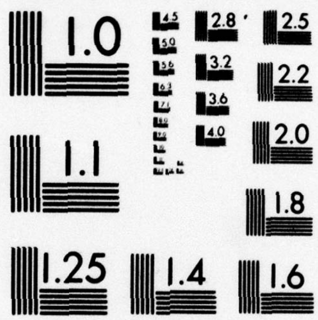
AD-A069 029 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/2
LAMAS SYSTEM MANUAL.(U)
FEB 78 DAAG39-77-C-0112
UNCLASSIFIED NL

1 OF 2
AD
A069 029


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

LEVEL II

TRW

ADVANCED STUDIES

ONE SPACE PARK · REDONDO BEACH · CALIFORNIA 90278

LAMAS SYSTEM MANUAL

CDRL ITEM A008

FEBRUARY 1978

CONTRACT NO. DAAG39-77-C-0112

Prepared for:
TRADOC Combined-Arms Test Activity
U.S.A. Intelligence Center and School

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

TRW.
DEFENSE AND SPACE SYSTEMS GROUP

ADVANCED STUDIES
ONE SPACE PARK · REDONDO BEACH · CALIFORNIA 90278

6

LAMAS SYSTEM MANUAL

CDRL ITEM A008

11 FEB[redacted] 1978

12 139p.

15 CONTRACT NO. DAAG39-77-C-0112

Prepared for:
TRADOC Combined–Arms Test Activity
U.S.A. Intelligence Center and School

409637

79 04 09 147

LAMAS SYSTEM MANUAL

This document describes the LAMAS software package from both a programmer and user standpoint. The text is written in Program Design Language (PDL) format and represents the final version of the working document used throughout program development. All LAMAS software and manual procedures are described, with the exception of the display software developed using company funds. A proprietary version of this document which describes both deliverable and company-funded software is maintained on file in the TRW Advanced Studies office.

TRW SYSTEMS GROUP
ONE SPACE PARK
REDONDO BEACH, CALIFORNIA 90278
(213) 535-4321

LOCATION AND MOVEMENT ANALYSIS SYSTEM

(LAMAS)

SOFTWARE TO COMPUTE GROUND FORCE MOVEMENT

CONSIDERING MOVEMENT DOCTRINE, ROAD CONDITIONS,

AND RISK FACTORS

21 FEB 78

PDL 03.06

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

```
************************
*                      *
* GENERAL DESCRIPTION  *
*                      *
************************
```

OVERVIEW

THE PURPOSE OF THIS SOFTWARE PACKAGE IS TO COMPUTE GROUND FORCE MOVEMENT USING OPTIMAL PATH DETERMINATION AND ROUTE CONFLICT RESOLUTION ALGORITHMS. TO INSURE FLEXIBILITY, USER INTERACTION IS SUPPORTED DURING MOST PHASES OF THE SOFTWARE'S OPERATION.

THREE MAJOR AREAS ARE INVOLVED IN THE PROCESS; INITIALIZATION GROUND FORCE MOVEMENT CALCULATIONS, AND DISPLAY.

INITIALIZATION

INITIALIZATION IS THE PROCESS OF PREPARING DATA BASES FOR GENERAL USE. THERE ARE THREE BASIC TYPES OF DATA BASES: PERMANENT, TEMPORARY - USER DEFINED, AND TEMPORARY - PROGRAM DEFINED.

PERMANENT DATA BASES ARE CREATED BEFORE THE USER EVER INVOKES THE SYSTEM, AND ARE GENERALLY UNALTERABLE BY THE USER. INCLUDED ARE SUCH FILES AS NODES, ROADS, CITIES, CROSS-COUNTRY MOVEMENT, CONCEALMENT, MAP←DIRECTORY, ROAD←DIRECTORY, AND CROSS-COUNTRY DIRECTORY.

A USER-DEFINED TEMPORARY DATA BASE IS A FILE OVER WHICH THE USER HAS CONSIDERABLE CONTROL, BUT WHICH EXISTS ONLY FOR A PARTICULAR SESSION ON THE SYSTEM. ONCE THE USER SIGNS OFF, THE DATA BASE IS LOST. THE UNITS DATA BASE IS THE PRIMARY EXAMPLE OF THIS TYPE IN LAMAS. HERE THE USER DEFINES THE CONTENTS, AND MAY ALTER THE CONTENTS AT ANY TIME, BUT THE FILE DOES NOT CARRY OVER FROM ONE SESSION TO ANOTHER.

PROGRAM-DEFINED TEMPORARY DATA BASES ARE FILES OVER WHICH THE USER HAS NO CONTROL WHATSOEVER. THE FILES MAY ENABLE THE USER TO DISPLAY SOME RESULTS, BUT NO ALTERATIONS MAY BE MADE. EXAMPLES OF THIS TYPE OF DATA BASE ARE SOLUTION←VECTORS, ROUTE←VECTORS, MAP←DIRECTORY IN MEMORY, AND ROAD←DIRECTORY IN MEMORY.

THE EFFECT OF INITIALIZATION VARIES FROM FILE TO FILE. FOR SOME, IT IS JUST SETTING ALL ELEMENTS TO ZERO, AS IN THE CASE OF THE SOLUTION←VECTORS AND ROUTE←VECTORS. FOR OTHERS, IT IS THE ESTABLISHING OF VALUES, AS IN ALL OF THE PERMANENT FILES, UNITS DATA BASE, AND THE DIRECTORIES.

THE MANNER IN WHICH THE VALUES ARE SET ALSO VARIES FROM FILE TO FILE. ALL PERMANENT FILES ARE CREATED BY TRANSFERRING INFORMATION FROM DATA CARDS TO DISK STORAGE. THE UNITS FILE IS ESTABLISHED BY DIRECT USER INPUT, WHILE THE NODE←VECTOR FILE IN MEMORY AND THE DIRECTORIES IN MEMORY ARE CREATED BY THE PROGRAM, BUT ONLY AFTER THE USER HAS INVOKED PARTICULAR ROUTINES.

GROUND MOVEMENT CALCULATIONS

THERE ARE TWO BASIC TYPES OF GROUND MOVEMENT CALCULATIONS WE WILL MAKE IN THIS PACKAGE. ONE CONSIDERS THE ROAD NETWORK FOR THE AREA OF INTEREST AS REPRESENTED IN THE NODE←VECTOR DATA BASE, WHILE THE OTHER USES CROSS-COUNTRY MOVEMENT INFORMATION AS SUPPLIED BY THE ARMY ON ITS CCM AND CONCEALMENT MAPS FOR THE AREA OF INTEREST.

THE MAJOR DIFFERENCES BETWEEN THESE TWO APPROACHES ARE IN THEIR SCOPE OF APPLICATION, AND THE DATA BASES WHICH SUPPLY THE INFORMATION. USING THE ROAD NETWORK APPROACH, ONE MAY CONSIDER UP TO TEN 1:50000 MAPS FOR ANY COMPUTATION; WHILE THE CROSS-COUNTRY MOVEMENT MUST BE LIMITED TO AT MOST ONE 1:50000 MAP. THIS IS DUE TO THE MANNER IN WHICH THE TWO TYPES ARE MODELED.

BECAUSE OF THE DIFFERENCE IN INFORMATION BEING HELD IN THE DATA BASE REGARDING THE TWO TYPES OF MOVEMENT, THIS CALLS FOR TWO VERY DIFFERENT DATA BASE APPROACHES. THESE WILL BE EXPLAINED LATER ON IN THIS DOCUMENT.

ON THE OTHER HAND, THE ALGORITHM USED TO CALCULATE THE PATHS FOR EACH TYPE OF MOVEMENT IS VIRTUALLY IDENTICAL. THE ONLY DIFFERENCE LIES IN THE INTERPRETATION OF THE DATA BASES; THE ACTUAL MECHANICS OF THE PATH CALCULATIONS ARE THE SAME.

THE ALGORITHM FINDS THE 'BEST' PATH FOR A UNIT TO TRAVEL. 'BEST' IS A USER-DEFINED PARAMETER MADE UP OF TWO COMPONENTS, RISK AND TIME. RISK IS A VALUE ARRIVED AT BY CONSIDERING ROAD CONDITIONS, TERRAIN FACTORS, WHETHER OR NOT A BRIDGE IS PRESENT, AND WHETHER OR NOT A CITY IS PRESENT. TIME IS THE AMOUNT OF TIME IT TAKES A PARTICULAR UNIT TO TRAVEL FROM ONE NODE TO ANOTHER.

OTHER TYPES OF PATH CALCULATIONS MAY BE PERFORMED AS WELL. THE USER MAY FIND THE SECOND BEST PATH TO TRY TO ATTACK THE SENSOR TASKING PROBLEM, HE MAY FIND THE BEST NODE AT WHICH TO PERFORM INTERDICTION FOR DISRUPTION PURPOSES, OR HE MAY FIND THE BEST LONGEST PATH (A METHOD OF CALCULATING HOW QUICKLY A FORCE MAY BUILD UP IN A PARTICULAR AREA.

USER FORMAT

THE FORMAT FOR USER INTERACTION WILL BE OF A SIMPLE NATURE. WHEN STARTING, A BASIC MENU OF AVAILABLE FUNCTIONS WILL BE LISTED ON A DISPLAY SCREEN. THE USER WILL DECIDE WHICH ONE HE WISHES TO PURSUE, AND ENTER ITS CODE AT A TERMINAL. THIS MAY CAUSE DIRECT ACTION, OR CAUSE ANOTHER MENU TO BE DISPLAYED. THROUGHOUT THE OPERATION, WHETHER OR NOT A MENU APPEARS, THE USER WILL BE PROMPTED WITH INSTRUCTIONS REGARDING HOW TO PROCEED. IN THIS WAY, THE USER HAS 'HANDS-ON' CONTROL OF ANY AND ALL ACTIVITIES INVOLVED IN THE GROUND FORCE MOVEMENT COMPUTATIONS.

SYSTEM ARCHITECTURE

THIS SOFTWARE HAS BEEN DESIGNED TO OPERATE ON A PDP 11/45 COMPUTER HAVING 96K WORDS OF MEMORY, WITH ANY SINGLE TASK ABLE TO ACCESS AT MOST 32K WORDS. THE OPERATING SYSTEM, RSX-11M, SUPPORTS A USER DEFINED MULTITASKING CAPABILITY, WITH THE TOTAL AMOUNT OF MEMORY USAGE TO NOT EXCEED 64K WORDS.

FOUR DIFFERENT PERIPHERALS ARE NEEDED TO EFFECTIVELY USE THIS SOFTWARE PACKAGE. ONE IS A TERMINAL FOR COMMUNICATION WITH THE PROGRAM. THIS MAY BE A DECWRITER, A TEKTRONIX CRT, OR A VT05 TERMINAL. ALL PERMANENT DATA BASES ARE STORED ON AN RP04 MOVING HEAD DISK UNIT, AS WELL AS ALL OF THE SOURCE AND OBJECT CODE, AND TASK IMAGE. A GOULD 4800 HIGH-SPEED LINE PRINTER IS USED FOR PATH STATISTICS OUPUT, AND A COMTAL 8500 IMAGE PROCESSING SYSTEM IS USED FOR THE VISUAL DISPLAYS.

DUE TO MEMORY CONSIDERATIONS, SUCH AS THE LARGE AMOUNT OF SPACE NEEDED FOR DATA BASES, IT HAS BEEN DECIDED TO USE MULTIPLE TASKING TO IMPLEMENT THIS PROGRAM. ONE TASK PERFORMS ALL OF THE I/O FUNCTIONS, THE OBTAINING OF USER INPUT AND THE PRINTING OF MESSAGES, WHILE ANOTHER DOES ALL OF THE COMPUTATION. RSX-11M IS STRUCTURED SO THAT THIS MAY BE DONE WITH A MINIMUM OF DELAY TIME USED TO SWITCH FROM ONE TASK TO THE OTHER. MESSAGES MAY BE SENT BETWEEN THE TWO, AND THIS IS HOW OUTPUT DATA IS TRANSFERRED. THERE IS ALSO A SEPARATE GRAPHICS PACKAGE WHICH IS ITSELF ANOTHER TASK, SO THERE ARE THREE TASKS RUNNING CONCURRENTLY IN THIS SYSTEM.

THE TOTAL MEMORY OF THESE THREE TASKS MUST NOT EXCEED 64K WORDS, OR SEVERE DEGRADATION OF RESPONSE TIME IS INCURRED. THIS NECESSITATES THE USE OF OVERLAYS IN ORDER TO SAVE MEMORY SPACE.

WE HAVE ARRIVED AT THESE ESTIMATES FOR STORAGE REQUIREMENTS:

DESCRIPTION	CONTENTS	MEMORY USED
IN+CORE+NODE+ARRAY	400 NODES AT 32 WORDS EACH (ABOUT 10 MAPS)	12,800 WORDS
SOLUTION+VECTORS	40 ROUTES WITH 25 NODES PER ROUTE.	5,000 WORDS

```
OTHER                    DIRECTORIES, COUNTERS, ETC.          5,000 WORDS
.
TOTAL                                                        22,800 WORDS
.
.
     THUS APPARENTLY LEAVING US WITH 10,000 WORDS FOR ACTUAL EXECUTABLE
     CODE PER OVERLAY.  HOWEVER, THE GRAPHICS PACKAGE USES 21,300 WORDS,
     AND THE I/O ROUTINES USE APPROXIMATELY 17,000 WORDS, SO ONLY ABOUT
     4,000 WORDS ARE TRULY AVAILABLE FOR CODING IN THE MAIN ROUTINE.
.
TIMINGS:     TO BE DETERMINED
```

```
**************************
*                        *
* DATA BASE DESCRIPTIONS *
*                        *
**************************
```

NODES AND LINKS

CONSISTING OF ONLY NODE+VECTORS: THIS DATA BASE IS THE LARGEST, AND PROBABLY THE MOST IMPORTANT, OF ALL THE DATA BASES. THE INFORMATION CONTAINED WITHIN ENABLES THE PATH ALGORITHM TO ACCURATELY CALCULATE THE BEST PATH FOR A GIVEN UNIT TO TRAVEL.

A NODE+VECTOR IS A 32-WORD STORAGE AREA WHICH CONTAINS DATA PERTAINING TO A NODE AND THE LINKS WHICH CONNECT IT TO OTHER NODES. A NODE IS A POINT OF INTEREST, USUALLY, BUT NOT NECESSARILY, BEING AT THE JUNCTION OF TWO OR MORE ROADS. THE LINK BETWEEN TWO NODES IS MOST OFTEN A REPRESENTATION OF THE ROADWAY JOINING THE TWO, BUT IT CAN ALSO BE A REPRESENTATION OF SIMPLY THE GENERAL PATH (CROSS-COUNTRY OR OTHERWISE) WHICH CONNECTS THEM. TWO NODES ARE SAID TO BE ADJACENT TO EACH OTHER IF THERE EXISTS A SINGLE LINK BETWEEN THEM. A PATH IS THEN A SERIES OF LINKS SUCH THAT THE LINKS, WHEN TAKEN AS A WHOLE, ARE CONNECTED. THAT IS, STARTING AT NODE 1, A LINK CONNECTS WITH NODE 2 ADJACENT TO NODE 1, WHICH CONNECTS WITH NODE 3 ADJACENT TO NODE 2, ETC., UNTIL THE DESTINATION NODE IS REACHED. ALL NODE-VECTOR DATA HAS BEEN OBTAINED FROM STANDARD L SERIES 1:50000 MAPS.

A GROUND FORCE UNIT'S MOVEMENT THROUGH A ROAD NETWORK IS CALCULATED BY MODELING ITS TRAVEL FROM ONE NODE TO ANOTHER ALONG A LINK. A LINK IS A CHARACTERIZATION OF A PATH, AND AS SUCH, PASSES ALONG ASSOCIATED PIECES OF INFORMATION, SUCH AS; DISTANCE, NUMBER OF LANES, ROAD TYPE, OFF-ROAD TRAFFICABILITY, AND TERRAIN, BRIDGE, AND CITY CODES.

DISTANCE REFERS TO THE ACTUAL DISTANCE IN KILOMETERS ONE WOULD HAVE TO TRAVEL TO MOVE FROM ONE NODE TO ANOTHER ALONG THE TRUE CONNECTION. NUMBER OF LANES MEANS THE NUMBER OF SINGLE-FILE ROAD LANES WHICH ARE EQUIVALENT TO THE LINKING ROAD. ROAD TYPE IS A CODE REFLECTING ANY OF FOUR POSSIBILITIES; AUTOBAHN OR AUTOSTRAßE, MAIN ROAD, SECONDARY ROAD, OR FAIR WEATHER ROAD ONLY. OFF-ROAD TRAFFICABILITY IS ANOTHER CODE, THIS TIME INDICATING OPEN TRAFFICABLE AREAS NEAR THE ROAD WHICH WILL ALLOW REST OR ALTERNATE ROUTES. TERRAIN MEASURES LAND FEATURES AND INTERVISIBILITY. THE CHOICES ARE FLAT, HILLY, MOUNTAINOUS, OR ANY OF THESE THREE PLUS HIGH TERRAIN IMMEDIATELY TO THE WEST. THE BRIDGE CODE INDICATES THE PRESENCE, OR LACK OF, A BRIDGE, AND IF IT EXISTS, ITS SIZE, EITHER SMALL, MEDIUM, OR LARGE. FINALLY, CITY CODE FUNCTIONS IN A MANNER IDENTICAL TO THE BRIDGE CODE. RISK IS TAKEN TO BE A LINEAR COMBINATION OF OFF-ROAD TRAFFICABILITY, TERRAIN, BRIDGE,

AND CITY CODES.

THE CROSS-COUNTRY AND CONCEALMENT PATH ALGORITHM DOES NOT USE THE NODES AND LINKS DATA BASE.

THE ELEMENTS WHICH CHARACTERIZE A NODE ARE SIGNIFICANTLY DIFFERENT THAN THOSE CHARACTERIZING A LINK. ALL NODAL INFORMATION IS LOCATION ORIENTED, THAT IS, IT DESCRIBES EXACTLY WHERE THE NODE IS ON THE MAP. THIS INFORMATION INCLUDES LATITUDE, LONGITUDE, THE MAP NUMBER FOR THE PARTICULAR MAP ON WHICH THE NODE RESIDES, AND THE NODE NUMBER (AN ARB-ITRARY FIGURE ESTABLISHED DURING THE MANUAL CREATION OF THE DATA BASE.)

A NODE←VECTOR THEN, IS COMPRISED OF THE ABOVEMENTIONED NODAL AND LINK INFORMATION (A MAXIMUM OF FOUR ADJACENT NODES MAY EXIST), PLUS SEVERAL OTHER VALUES, WHOSE SIGNIFICANCE WILL BECOME CLEAR LATER ON IN THIS DOCUMENT (AMONG THESE ARE WORTH MEASURE, PREDECESSOR NODE, CUM-MULATIVE TIME, CUMMULATIVE WORTH MEASURE, TIME MEASURE, SOLUTION VECTOR POINTER, LANE USED, RISK MEASURE, AND PARK TIME. THESE ARE ALL USED BY THE PATH ALGORITHM.)

NODE←VECTORS MAKE UP A PERMANENT DATA BASE WHICH RESIDES ON DISK. ONE MAY THINK OF THIS STORAGE AS BEING DIVIDED INTO BINS, SUCH THAT ONE BIN CONTAINS THE NODE←VECTORS FOR ONE MAP. A DIRECTORY IS ALSO STORED ON DISK, AND IT CONTAINS IDENTIFICATION FOR EACH BIN, THAT IS, THE AP-PROPRIATE MAP NUMBER. ACCESS OF ANY MAP'S NODE←VECTORS IS QUITE SIMPLE; ONE FINDS THE MAP NUMBER IN THE DIRECTORY. THE DIRECTORY ENTRY POINTS TO THE BIN ON DISK WHICH CONTAINS THE NODE←VECTORS, WHICH MAY THEN BE READ INTO MAIN MEMORY.

ONCE IN MAIN MEMORY, NODE←VECTORS ARE ALTERABLE BY THE USER. FUNCTIONS EXIST WHICH ALLOW THE USER TO CHANGE CERTAIN VALUES, SUCH AS LINK DISTANCE, ROAD TYPE, OR NUMBER OF LANES. ANY CHANGES MADE ARE NOT PERMANENT, SINCE THE "CORRECTED" VERSION OF THE DATA BASE IS NEVER COPIED ONTO THE PERMANENT DISK FILE.

WHILE IN MEMORY, THE NODE←VECTORS ARE THOUGHT OF AS OCCUPYING A NODE←VECTOR ARRAY. MAIN MEMORY IS LINEARLY ORGANIZED, SO IT IS NAT-URAL TO THINK OF NODE←VECTORS COMPRISING A LIST INDEXABLE BY INTEGERS. THUS, IF TWO MAPS' NODE←VECTORS ARE IN MEMORY, WITH THE FIRST MAP HAVING 30 NODE←VECTORS, THE FIRST NODE←VECTOR OF THE SECOND MAP IS THE 31 ST NODE←VECTOR IN THE ARRAY, ETC.

NODE VECTOR FORMAT

```
##########################################################################
#                                                                        #
#                 NOTE THAT EACH WORD IS SIXTEEN BITS,                   #
#                 AND IT IS OFTEN POSSIBLE TO CONSIDER TWO               #
#                 PIECES OF INFORMATION FOR ONE WORD, WITH               #
#                 EACH PIECE OCCUPYING ONE BYTE.  IF THERE               #
#                 IS NO SECOND INFORMATION PIECE IN A WORD,              #
#                 THE DATA IS RIGHT-ADJUSTED WITHIN THE WORD.            #
#   .                                                                    #
#    WORD #       BYTE 1                    BYTE 2                       #
#     1      NODE MAP+NUMBER                                             #
#     2      NODE # WITHIN MAP              ;LABEL                       #
#     3      WORTH MEASURE                                               #
#     4      PREDECESSOR NODE NUMBER                                     #
#     5      CUMMULATIVE TIME                                            #
#     6      CUMMULATIVE WORTH MEASURE                                   #
#     7      TIME MEASURE                                                #
#     8      SOLUTION-VECTOR POINTER                                     #
#     9      LATITUDE                                                    #
#    10      LONGITUDE                                                   #
#    11      NUMBER OF ADJACENT NODES       ;LANE USED                   #
#    12      ADJACENT MAP #1                                             #
#    13      NODE+NUMBER WITHIN MAP         ;LINK DISTANCE               #
#    14      # OF LANES                     ;ROAD TYPE                   #
#    15      OFF-ROAD TRAFFICABILITY        ;TEPRAIN                     #
#    16      BRIDGE                         ;CITY                        #
#    17      ADJACENT MAP # 2                                            #
#    18      NODE # WITHIN MAP              ;LINK DISTANCE               #
#    19      # OF LANES                     ;ROAD TYPE                   #
#    20      OFF-ROAD TRAFFICABILITY        ;TERRAIN                     #
#    21      BRIDGE                         ;CITY                        #
#    22      ADJACENT MAP #3                                             #
#    23      NODE # WITHIN MAP              ;LINK DISTANCE               #
#    24      # OF LANES                     ;ROAD TYPE                   #
#    25      OFF-ROAD TRAFFICABILITY        ;TERRAIN                     #
#    26      BRIDGE                         ;CITY                        #
#    27      ADJACENT MAP #4                                             #
#    28      NODE # WITHIN MAP              ;LINK DISTANCE               #
```

```
# 37    29    # OF LANES                  ;ROAD TYPE      #    251
# 38    30    OFF-ROAD TRAFFICABILITY     ;TERRAIN        #    252
# 39    31    BRIDGE                      ;CITY           #    253
# 40    32    RISK MEASURE                ;PARK TIME      #    254
#                                                         #
###########################################################
```

NODE VECTOR DESCRIPTION

WORD #	DESCRIPTION AND USE	RANGE
1	NUMBER OF 1:50000 MAP FROM WHICH DATA IS TAKEN.	1-32767
2,1	NUMBER OF NODE WITHIN THE MAP. THESE NUMBERS ARE SELECTED WHEN THE NODES ARE MANUALLY DE-FINED.	1-80
2,2	FLAG TO INDICATE WHETHER OR NOT THIS NODE HAS BEEN LABELED BY THE PATH ALGORITHM.	0-1
3	WORTH OF THE LINK BETWEEN THIS NODE AND ITS PREDECESSOR, AS DETERMINED BY THE PATH ALGORITHM.	0-100
4	NODE+VECTOR INDEX OF THE PREDECESSOR NODE+VEC-TOR, SELECTED BY THE OPTIMAL MOVE ALGORITHM.	1-400
5	THE CUMMULATIVE TIME, IN MINUTES, OF A MOVEMENT UNIT AT THIS NODE POSITION AS DETERMINED BY THE OPTIMAL MOVEMENT ALGORITHM. THERE IS A MAXIMUM TRAVEL TIME OF TEN DAYS (=14400 MINUTES) PLUS 23 HOURS, 59 MINUTES (=1439 MINUTES).	0-15839
6	TOTAL WORTH MEASURE OF THE PATH FROM INITIAL NODE UP TO AND INCLUDING THIS LINK'S WORTH MEASURE (WORD 3).	0-32767
7	THE ACTUAL TIME FOR A UNIT TO PROGRESS FROM PRE-DECESSOR TO THIS NODE.	0-15839
8	INDEX TO ENTRY IN THE SOLUTION+VECTOR ARRAY. THIS POINTER ENABLES THE NODE+VECTOR TO BE ASSOCIATED WITH THE OPTIMAL ROUTE DATA. ONLY NODES THAT ARE PART OF AN OPTIMAL SOLUTION WILL HAVE THIS ENTRY. IF THE NODE IS PART OF SEV-ERAL OPTIMAL ROUTES, THIS WORD WILL POINT TO THE	0-5000

	MOST RECENTLY CALCULATED SOLUTION+VECTOR.	
9	NODE LATITUDE IN KILOMETERS FROM EQUATOR. THIS DATA IS ONLY USED FOR DISPLAY AND IS THUS + OR - 1 KM RESOLUTION. ROUTE ALGORITHMS USE LINK DISTANCES, NOT THESE LATITUDE, LONGITUDE COORDINATES.	0-32767
10	NODE LONGITUDE IN KILOMETERS FROM GREENWICH. THIS DATA IS ONLY USED FOR DISPLAY AND IS THUS + OR - 1 KM RESOLUTION.	0-32767
11,1	NUMBER OF ADJACENT NODES, MAXIMUM IS FOUR, MINIMUM IS ONE.	1-4
11,2	LANE NUMBER USED DURING PATH CALCULATION. THIS IS NECESSARY IN ORDER TO EFFECTIVELY USE MULTIPLE LANED ROADS.	1-4
12	THE MAP+NUMBER OF THE FIRST ADJACENT NODE. THIS ENTRY WILL BE RESET AFTER DATA IS READ FROM DISK TO CORE TO BECOME THE NODE+VECTOR ARRAY INDEX OF THE FIRST ADJACENT NODE.	1-32767
13,1	NUMBER OF NODE WITHIN THE MAP OF THE FIRST ADJACENT NODE.	1-80
13,2	LINK DISTANCE IN TENTHS OF A KM BETWEEN THE NODE AND THE FIRST AJDACENT NODE.	0-255
14,1	NUMBER OF SINGLE FILE ROAD LANES BETWEEN THE NODE AND THE FIRST ADJACENT NODE.	1-4
14,2	ROAD TYPE CODE. 1=AUTOBAHN OR AUTOSTRADE, 2=MAIN ROAD, 3=SECONDARY ROAD OR ROAD, 4=FAIR WEATHER ROAD ONLY.	1-4
15,1	GROUND COVER CODE TO INDICATE OPEN TRAFFIC-	1-5

```
#                ABLE AREAS NEAR ROAD THAT ALLOWS REST OR ALTER-                         #
#                NATE PATHS.                                                             #
#                1=URBAN, 2=CULTIVATED, 3=WOODED, 4=SWAMP,                               #
#                5=ROAD NET ADJACENT TO MAIN LINK.                                       #
#                                                                                        #
#   15,2    TERRAIN.  CODE TO INDICATE TERRAIN AND COARSE        1-13                    #
#           INTERVISIBILITY MEASURE.  1=FLAT, 2-HILLY,                                   #
#           3=MOUNTAINOUS, 10=HIGH TERRAIN IMMEDIATELY TO THE                            #
#           WEST, AND ANY OF THE FIRST THREE PLUS 10.                                    #
#                                                                                        #
#   16,1    BRIDGE.0=NONE, 1=SMALL, 2=MEDIUM, 3=LARGE.           0-3                     #
#                                                                                        #
#   16,2    CITY.  0=NONE, 1=SMALL, 2=MEDIUM, 3=LARGE            0-3                     #
#                                                                                        #
#   17-21   IDENTICAL TO WORDS 12-16 BUT APPLICABLE TO THE                               #
#           SECOND ADJACENT NODE, IF PRESENT.                                            #
#                                                                                        #
#   22-26   IDENTICAL TO WORDS 12-16 BUT APPLICABLE TO THE                               #
#           THIRD ADJACENT NODE, IF PRESNET.                                             #
#                                                                                        #
#   27-31   IDENTICAL TO WORDS 12-16 BUT APPLICABLE TO THE                               #
#           FOURTH ADJACENT NODE, IF PRESENT.                                            #
#                                                                                        #
#   32,1    RISK MEASURE FOR THE LINK BETWEEN THIS NODE AND      0-127                   #
#           ITS PREDECESSOR.  RISK IS TAKEN TO BE THE SUM OF                             #
#           GROUND COVER, TERRAIN, BRIDGE, AND CITY CODES.                               #
#                                                                                        #
#   32,2    USED BY THE PATH ALGORITHM TO REMEMBER HOW MANY      0-255                   #
#           MINUTES A UNIT PARKED AT THIS NODE.                                          #
#                                                                                        #
##########################################################################################
****** TOO MANY LINES IN SEGMENT
```

TERRAIN

THIS DATA BASE SUPPORTS THE TERRAIN MODEL FOR PATH CALCULATIONS. THERE ARE ACTUALLY THREE DATA BASES INVOLVED, CROSS-COUNTRY MOVEMENT, SUMMER CONCEALMENT, AND WINTER CONCEALMENT.

EACH DATA BASE SUPPLIES DATA FOR A SPECIFIC TYPE OF PATH CALCULATION. CROSS-COUNTRY YIELDS DATA FOR FASTEST MOVEMENT, AND THE TWO CONCEALMENT DATA BASES GIVE INFORMATION ABOUT THE RISK OF A PATH.

THESE DATA BASES WERE CREATED BY PLACING A 93 X 88 SQUARE GRID OVER THE DATA MAPS SUPPLIED BY THE ARMY. EACH SQUARE WAS THEN ASSIGNED A VALUE CORRESPONDING TO THE AREA WHICH THE SQUARE COVERED. THESE VALUES WERE TRANSFERRED TO COMPUTER CARDS WHICH WERE THEN USED TO CREATE THREE DATA FILES ON DISK, 'CCM.DAT', 'SCON.DAT', AND 'WCON.DAT'.

BECAUSE OF THE REGULARITY OF THE SQUARES, EACH SQUARE MAY BE REGARDED AS A NODE WITH EIGHT NEIGHBORS (UNLESS IT LIES ON AN EDGE). THUS, EACH MAP IS REPRESENTED BY 8184 NODES. FOR ANY NODE, NO MORE INFORMATIONS IS NEEDED, SINCE ALL LINK ATTRIBUTES ARE KNOWN.

THERE IS, HOWEVER, PATH INFORMATION WHICH MUST BE STORED DURING A CALCULATION, BUT THERE SIMPLY ISN'T ENOUGH MEMORY TO ALLOCATE 5 WORDS FOR EACH OF THE 8184 NODES. THUS, WHEN USING THE TERRAIN MODEL FOR PATH CALCULATIONS, THERE IS AN ADDITIONAL DATA BASE CALLED THE 'WORKING LIST'. AS A NEW NODE IS INCLUDED IN THE CALCULATION, A NEW ENTRY TO THE WORKING LIST IS MADE. SINCE NOT ALL NODES WILL BE INCLUDED, THE WORKING LIST'S SIZE IS CONSIDERABLY SMALLER THAN 8184 * 5 WORDS, AND FITS IN MAIN MEMORY WITH EASE.

ANOTHER SPACE-SAVING DEVICE IS EMPLOYED. THE CODE FOR ANY SQUARE IS NEVER GREATER THAN 15, SO DATA FOR FOUR NODES MAY BE PLACED IN ONE WORD OF MEMORY, THUS 8184 CODES USE ONLY 2046 WORDS.

CROSS-COUNTRY AND CONCEALMENT CODE DESCRIPTIONS

THESE CODES COME DIRECTLY FROM THE MAPS PREPARED BY THE TERRAIN ANALYSIS CENTER, US ARMY ENGINEER TOPOGRAPHIC LABORATORIES, FORT BELVOIR, VIRGINIA, IN SEPTEMBER, 1977.

CROSS-COUNTRY MOVEMENT

CODE	ESTIMATED MAX. SPEED OF XM-1 TANK (MPH)	MANEUVERABILITY DESCRIPTOR
1	>= 25	EXCELLENT
2	20 - 25	VERY GOOD
3	10 - 20	GOOD
4	5 - 10	FAIR
5	< 5	POOR
6	PASSAGE BLOCKED	NO GO
7	BUILT-UP AREA (TRAVEL OFF ROADS NOT EVALUATED)	

CONCEALMENT

CODE	PERCENT CHANCE OF BEING OBSERVED FROM THE AIR
8	0 - 25 (BEST CONCEALMENT)
9	25 - 50
10	50 - 75
11	75 - 100 (POOREST CONCEALMENT)

UNITS

THIS IS A TEMPORARY DATA BASE, THAT IS, A DATA BASE WHICH MAY CHANGE FROM USER TO USER, AND, IN FACT, FROM ONE PATH CALCULATION TO ANOTHER, IF DESIRED. BECAUSE OF ITS TEMPORARY NATURE, IT EXISTS ONLY IN MAIN MEMORY; THERE IS NO UNITS DATA BASE ON DISK.

THE UNIT←VECTOR IS THE SOLE CONSTITUENT WITHIN THIS DATA BASE. IT CONTAINS INFORMATION PERTAINING TO A UNIT, SUCH AS ITS NAME, STARTING LOCATION, DESTINATION LOCATION, PRESENT LOCATION, PRIORITY, START TIME, AND TYPE CODE. EACH LOCATION IS ENTERED BY THE USER AS A MAP←NUMBER, NODE←NUMBER PAIR, WHERE THE MAP←NUMBER IS THE NUMBER OF THE 1:50000 MAP ON WHICH THE NODE LIES, AND THE NODE←NUMBER IS THAT NUMBER ARBITRARILY ASSIGNED WHEN THE NODE←VECTOR DATA BASE WAS CONSTRUCTED. HOWEVER, FOR PURPOSES OF STORAGE, THESE LOCATIONS ARE REPRESENTED INTERNALLY AS INDEXES INTO THE NODE←VECTOR ARRAY. PRIORITY IS A NUMBER REPRESENTING A UNIT'S IMPORTANCE WITH RESPECT TO OTHER UNITS. START TIME IS A VALUE ENTERED BY A PATH ALGORITHM, NOT THE USER, AND REPRESENTS MINUTES OF TIME. TYPE CODE IS A NUMBER WHICH STANDS FOR THE PARTICULAR TYPE OF UNIT BEING CONSIDERED, SUCH AS A RIFLE BATTALION, OR A MOTORIZED DIVISION.

THIS DATA BASE MAY BE RECREATED EACH TIME A USER WISHES TO PERFORM A PATH CALCULATION, AND MAY CONTAIN ANYWHERE FROM 1 TO 60 SEPARATE UNITS. IF NO UNITS ARE ESTABLISHED BY THE USER, THE PATH ALGORITHMS WILL NOT FUNCTION, AND AN ERROR MESSAGE PRINTED.

UNIT+VECTOR FORMAT

WORD #	BYTE 1	BYTE 2
1	FIRST LETTER	SECOND LETTER
2	THIRD LETTER	FOURTH LETTER
3	FIFTH LETTER	SIXTH LETTER
4	SEVENTH LETTER	EIGHTH LETTER
5	STARTING LOCATION	
6	DESTINATION LOCATION	
7	PRESENT LOCATION	
8	PRIORITY	TYPE CODE
9	START TIME	
10	SPARE	

UNIT←VECTOR DESCRIPTION

WORD #	DESCRIPTION AND USE	RANGE
1-4	THIS IS THE UNIT'S NAME IN CHARACTER FORMAT. THERE MAY BE AT MOST EIGHT CHARACTERS.	ANY ALLOWABLE CHARACTER
5	AN INDEX INTO THE NODE←VECTOR ARRAY. THUS, THIS WORD SAYS THAT IF THE INDEX EQUALS I, THE PROPER NODE IS THE I TH NODE IN THE ARRAY.	1-400
6	AN INDEX INTO THE NODE←VECTOR ARRAY, INDICATING THE DESTINATION OF THIS UNIT.	1-400
7	AN INDEX INTO THE NODE←VECTOR ARRAY, INDICATING WHERE THE UNIT IS AT THIS MOMENT IN THE CALCULATION.	1-400
8,1	PRIORITY OF THIS UNIT. 1 IS THE HIGHEST, 127 IS THE LOWEST. 0 INDICATES THAT THE PRIORITY HAS NOT BEEN ESTABLISHED.	0-127
8,2	CODE INDICATING GENERAL TYPE OF UNIT.	0-255
9	TIME AT WHICH UNIT IS KNOWN TO BE AT EITHER ITS STARTING OR FINISHING NODE. THIS DEPENDS UPON MOVEMENT TYPE.	0-15839
10	SPARE	

SOLUTIONS

ANOTHER TEMPORARY DATA BASE, THIS IS CREATED BY THE PATH ALGORITHM AS IT RECONSTRUCTS A ROUTE OF TRAVEL. EACH SOLUTION←VECTOR SO CREATED CONTAINS INFORMATION ABOUT A PARTICULAR NODE; ITS NODE←VECTOR INDEX, TIME IN, TIME OUT, LANE NUMBER, AND PARK TIME, AND FURTHER DIAGNOSTIC DATA; A POINTER TO ANOTHER SOLUTION←VECTOR WHICH CONSIDERS THE SAME NODE←VECTOR (IF NONE, THE POINTER IS NULL), AND WHETHER OR NOT THIS SOLUTION←VECTOR SHOULD BE USED BY THE PATH DECONFLICTING LOGIC.

A NODE←VECTOR INDEX IS A POINTER TO A NODE←VECTOR IN MAIN MEMORY. TIME IN AND TIME OUT INDICATE A TIME PERIOD DURING WHICH THE NODE←VECTOR IS BUSY. LANE NUMBER KEEPS TRACK OF WHICH LANE THIS UNIT USED TO TRAVEL THROUGH THIS NODE. THIS ENABLES MULTIPLE LANE ROADS TO BE USED BY MORE THAN ONE UNIT AT A TIME. PARK TIME SHOWS HOW MUCH TIME (IN MINUTES) THE UNIT HAD TO WAIT AT THIS NODE'S PREDECESSOR BEFORE TRAVELING TO THIS NODE.

THE SOLUTION←VECTOR POINTER ALLOWS THE DECONFLICTING LOGIC TO OPERATE. BY FOLLOWING THE POINTERS, IT IS POSSIBLE TO FIND ALL INTERVALS DURING WHICH A PARTICULAR NODE IS BUSY. IN THIS WAY, DECONFLICTING, THE PROCESS OF MAKING SURE THAT NO TWO UNITS USE A NODE DURING OVERLAPPING TIME INTERVALS, MAY BE ACCOMPLISHED. THE USER HAS THE OPTION TO INCLUDE OR EXCLUDE THE DECONFLICTING LOGIC. DATA WITHIN THE SOLUTION←VECTOR REMEMBERS THE USER'S CHOICE.

THE ROUTE DESCRIBED BY A PATH IS A SERIES OF SOLUTION←VECTORS HAVING A ONE-TO-ONE CORRESPONDENCE WITH THE NODES OF THE PATH. IF A PATH'S NODES WERE NUMBERED 1-N, ITS SOLUTION←VECTORS WOULD BE NUMBERED M+1-M+N, ASSUMING M SOLUTION←VECTORS ALREADY EXISTED AT THE TIME THE PATH WAS CALCULATED. NODE 1 WOULD CORRESPOND TO SOLUTION←VECTOR M+1, NODE 2 WOULD CORRESPOND TO SOLUTON←VECTOR M+2, ETC.

SOLUTION-VECTOR FORMAT

WORD #	CONTENTS
1	NODE NUMBER
2	TIME IN
3	TIME OUT
4	MULTIPLE USE NODE POINTER TO ENTRY IN THIS LIST.
5	+,- LANE NUMBER;PARK TIME

SOLUTION←VECTOR DESCRIPTION

WORD #	DESCRIPTION AND USE	RANGE
1	INDEX NUMBER OF THE ASSOCIATED NODE←VECTOR. SEVERAL ROUTES FOR DIFFERENT UNITS MAY POINT TO THE SAME NODE←VECTOR.	1-32767
2	THIS IS THE SIMULATED TIME THAT THE UNIT ENTERS THE NODE AS DETERMINED DURING CONSTRUCTION OF THE SOLUTION←VECTOR. TIME UNITS ARE MINUTES.	0-14400 MIN.
3	THIS IS THE SIMULATED TIME THAT THE UNIT LEAVES THE NODE. IT IS COMPUTED AS A COMPANION WITH THE ABOVE WORD #2. THE UNITS ARE IDENTICAL.	0-14400 MIN.
4	THIS NODE←ARRAY INDEX POINTS TO ANOTHER ENTRY IN THIS SOLUTION VECTOR LIST WHICH IS ASSOCIATED WITH THE SAME NODE←VECTOR. THIS LINK SHOWS THE MULTIPLE USE OF A NODE BY SEVERAL ROUTES OF DIFFERENT UNITS. CHECKING THE TIMES IN AND OUT (WORDS 2 AND 3) OF THE MULTIPLE USERS OF THE NODE WILL INDICATE WHETHER OR NOT A CONFLICT HAS OCCURRED.	0-32767
5,1	THIS IS THE LANE NUMBER WHICH THIS UNIT USED TO TRAVEL THROUGH THIS NODE.	1-4
5,2	TIME IN MINUTES SPENT PARKING AT THE PREDECESSOR NODE BEFORE TRAVELING TO THIS NODE. A POSITIVE VALUE INDICATES THE VECTOR SHOULD BE CONSIDERED WHEN DE-CONFLICTING. IF NEGATIVE, IT MUST BE SKIPPED.	+,- 0-32767 MI

ROUTES

A PATH HAS BEEN DEFINED TO BE A SERIES OF NODES THROUGH WHICH A UNIT TRAVELS IN ORDER TO GET FROM ONE SPECIFIC NODE TO ANOTHER. A ROUTE←VECTOR CONTAINS INFORMATION REGARDING A PARTICULAR ROUTE. THIS INCLUDES A POINTER TO THE UNIT WHICH IS TRAVELING ALONG THE PATH, A POINTER TO THE FIRST SOLUTION←VECTOR FOR THE ROUTE, TOTAL MARCH TIME, TOTAL RISK MEASURE FOR THE PATH, TOTAL PATH DISTANCE, NUMBER OF SOLUTION←VECTORS IN THE ROUTE, TYPE OF MOVEMENT (FORWARD OR BACKWARD IN TIME), AND INITIAL TIME.

THE UNIT POINTER IS AN INDEX INTO THE UNITS DATA BASE, THE SOLUTION←VECTOR POINTER IS AN INDEX INTO THE SOLUTIONS DATA BASE, AND INITIAL TIME IS EITHER THE START TIME OR FINISH TIME, AS DICTATED BY THE MOVEMENT TYPE.

A ROUTE←VECTOR IS CREATED IMMEDIATELY AFTER THE SOLUTION←VECTORS ARE GENERATED FOR A PARTICULAR ROUTE. THE ROUTES DATA BASE IS THEN TEMPORARY, AND EXISTS ONLY IN MAIN MEMORY. THE USER HAS NO CONTROL OVER THE DATA PLACED WITHIN A ROUTE←VECTOR, AND CAN ONLY KNOW ITS CONTENTS THROUGH INDIRECT MEANS. ONE ROUTINE LISTS UNIT NAMES AND ASSOCIATED ROUTES. ANOTHER LISTS PATH DIAGNOSTICS, SUCH AS TOTAL TRAVEL TIME, TOTAL DISTANCE, AND TOTAL RISK MEASURE, ALONG WITH OTHER INFORMATION. EACH OF THESE USES THE ROUTES DATA BASE AS A SOURCE.

ROUTE<VECTOR FORMAT

WORD #	CONTENTS
1	UNIT NUMBER
2	LIST HEAD POINTER
3	TOTAL ROUTE TIME
4	TOTAL ROUTE RISK MEASURE
5	TOTAL ROUTE DISTANCE
6	NUMBER OF SOLUTION<VECTORS IN ROUTE
7	MOVEMENT TYPE
8	INITIAL TIME

ROUTE←VECTOR FORMAT DESCRIPTION

WORD #	DESCRIPTION AND USE	RANGE
1	INDEX TO UNIT DATA BASE, SO THAT WE KNOW WHICH UNIT USES THIS ROUTE.	1-60
2	INDEX TO THE SOLUTION VECTOR LIST, POINTING TO THE FIRST SOLUTION VECTOR WHICH HAS THIS ROUTE NUMBER.	1-5000
3	TOTAL ELAPSED TIME OF MARCH FROM STARTING NODE TO DESTINATION NODE, INCLUDING RESTS AND STOPS, IN MINUTES.	0-15839 MIN.
4	TOTAL CUMMULATIVE RISK MEASURE OF ROUTE	0-32767
5	TOTAL ROUTE DISTANCE, IN TENTHS OF KMS.	0-3276.7
6	TOTAL NUMBER OF NODES VISITED	1-400
7	CODE INDICATING THAT THIS WAS DETERMINED USING A STARTING TIME (=0) OR AN ARRIVAL TIME (=1).	0 OR 1
8	TIME, IN MINUTES, WHICH THE USER DESIGNATED AS EITHER A STARTING TIME OR AN ARRIVAL TIME. DICTATED BY MOVEMENT TYPE.	0-15839

MAP DIRECTORY

THIS DATA BASE KEEPS TRACK OF THE MAPS DATA BASE. THERE IS ONE ENTRY FOR EACH MAP IN THE DISK FILE. THIS ENTRY CONTAINS TWO PIECES OF INFORMATION; 1) MAP NUMBER, AND 2) NUMBER OF NODES IN THE MAP. THE FIRST ENTRY IN THE MAP DIRECTORY CONTAINS A VALUE EQUAL TO THE NUMBER OF MAPS WHICH HAVE BEEN ENTERED. WHEN A MAP'S NODE INFORMATION IS PLACED IN THE DATA BASE, ITS DIRECTORY ENTRY IS PLACED IN THE NEXT AVAILABLE LOCATION OF THE MAP DIRECTORY. THUS THE SECOND MAP DIRECTORY ENTRY CORRESPONDS TO THE FIRST MAP PLACED IN THE MAPS DATA BASE, THE THIRD ENTRY CORRESPONDS TO THE SECOND MAP, ETC.

CROSS-COUNTRY DIRECTORY

THIS DIRECTORY IS USED TO FIND SPECIFIC MAPS IN THE CROSS-COUNTRY AND CONCEALMENT DISK FILES. EACH MAP HAS ONE DIRECTORY ENTRY, THIS ENTRY CONTAINING THE MAP NUMBER, AND THE UTM LATITUDE AND LONGITUDE EXTREMES OF THE MAP. THESE ARE THE COORDINATES OF THE LOWER LEFT AND UPPER RIGHT CORNERS. THE FIRST ENTRY OF THE DIRECTORY CONTAINS ONLY A COUNTER WHICH KEEPS TRACK OF HOW MANY ENTRIES ARE PRESENT IN THE DIRECTORY.

```
***********************
*                     *
* PREPARATION FOR USE *
*                     *
***********************
```

PREPARATION

```
REF
PAGE ************************************************************************
     *                                                                      *
     * 1   do ONCE BEFORE MAIN+PROGRAM EXECUTION                            *     626
  27 * 2       CREATE NODE+VECTOR AND MAP+DIRECTORY FILES ON DISK           *     627
  33 * 3       CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK *   628
     * 4   enddo ONCE BEFORE MAIN-PROGRAM EXECUTION                         *     629
     * 5   do AS REQUIRED                                                   *     630
     * 6       AMEND FILES                                                  *     631
     * 7   enddo AS REQUIRED                                                *     632
     *                                                                      *
     ************************************************************************
```

CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK

```
REF
PAGE
     .BEFORE THE FILES MAY BE CREATED, IT IS NECESSARY TO CREATE
     .NODE AND LINK CARDS. INFORMATION FOR THESE CARDS IS VISUALLY
     .DETERMINED FROM THE PARTICULAR L-SERIES MAP BEING CONSIDERED.
     .
     do FOR EACH MAP
         do FOR EACH NODE
28           CREATE NODE CARD
         enddo FOR EACH NODE
         do FOR EACH LINK
30           CREATE LINK CARD
         enddo FOR EACH LINK
     enddo FOR EACH MAP
     .
     .NOW THAT THE CARDS ARE READY FOR PROCESSING, CREATE THE NODE←VECTOR
     .FILE ON DISK, THAT IS, ESTABLISH NODE←VECTORS USING INFORMATION GAINED
     .FROM NODE AND LINK CARDS.
     .FIRST, INITIALIZE THE FILES
     .
     do ONCE
         CREATE DISK FILE FOR NODE←VECTORS, 'MAPS.DAT'
         CREATE DISK FILE FOR MAP←DIRECTORY, 'MAPDIR.DAT'
     enddo ONCE
     .
     .NOW BUILD THE VECTORS IN MEMORY
     .
     do UNTIL ALL MAPS ARE PROCESSED
         do FOR EACH MAP
             do UNTIL NO MORE NODE CARDS
                 READ A NODE CARD
                 PLACE INFORMATION INTO MAIN MEMORY
             enddo UNTIL NO MORE NODE CARDS
             do UNTIL NO MORE LINK CARDS
```

```
              READ A LINK CARD... NODE A TO NODE B
              FILL IN LINK INFORMATION TO NODE+VECTOR FOR A
              if NODE+VECTOR FOR B IS ON THIS MAP
                  FILL IN LINK INFORMATION TO NODE+VECTOR FOR B
              else CYCLE
              endif NODE+VECTOR FOR B IS ON THIS MAP
          enddo UNTIL NO MORE LINK CARDS
          READ 'MAPDIR.DAT' INTO MEMORY
          READ NUMBER OF MAPS PROCESSED
          INCREMENT NUMBER OF MAPS PROCESSED
          CREATE MAP+DIRECTORY ENTRY FOR THIS MAP
          WRITE NODE+VECTOR ARRAY FOR THIS MAP TO 'MAPS.DAT'
          WRITE MAP+DIRECTORY TO 'MAPDIR.DAT'
      enddo FOR EACH MAP
  enddo UNTIL ALL MAPS ARE PROCESSED
```

CREATE NODE CARD

REF
PAGE

```
*************************************************************************************
*                                                                                   *
*  1   .NODE CARDS WILL BE KEY PUNCHED MANUALLY, USING THE FOLLOWING FORMAT FOR     *    682
*  2   .THEIR CONSTRUCTION.                                                         *    683
*                                                                                   *
*************************************************************************************
```

NODE CARD FORMAT

ENTRY	EXAMPLE	FIELD
NODE MAP NUMBER. THE 1:50000 MAPS WILL BE USED. ENTER THE NUMERICAL MAP NUMBER, NOT THE LETTERS.	4924	1-5(5)
NODE NUMBER WITHIN THE MAP. A MASTER OVERLAY (PLASTIC) FOR EACH MAP MUST BE KEPT TO KEEP TRACK OF THE NUMBERS AND NODES.	18	6-9(4)
NODE LATITUDE IN KM'S FROM THE EQUATOR, READ DIRECTLY FROM THE 1:50000 MAP.	3245	10-16(7)
NODE LONGITUDE IN KM'S FROM GREENWICH, READ DIRECTLY FROM THE 1:50000 MAP.	125	17-23(7)
FIRST ADJACENT NODE MAP #.	4923	24-28(5)
FIRST ADJACENT NODE NUMBER FOR THE MAP OF THIS ADJACENT NODE. ENTFR THE NUMBER ONLY.	201	29-32(4)
SECOND ADJACENT NODE DATA, ENTER THE NUMBER ONLY.		33-41(9)
THIRD ADJACENT NODE DATA, ENTER THE NUMBER ONLY.		42-50(9)
FOURTH ADJACENT NODE DATA, ENTER THE NUMBER ONLY.		51-59(9)

```
# 37   NUMBER OF ADJACENT NODES.          4          60-61(2)                          #      721
# 38                                                                                   #      722
#                                                                                      #
########################################################################################
```

CREATE LINK CARD

REF PAGE

```
************************************************************************
*                                                                      *
* 1   . LINK CARDS WILL BE KEY-PUNCHED MANUALLY, USING THE            *     724
* 2   .FOLLOWING FORMAT.                                               *     725
*                                                                      *
************************************************************************
```

LINK CARD FORMAT

ENTRY	EXAMPLE	FIELD
FIRST NODE MAP NUMBER.	4924	1-5(5)
FIRST NODE NUMBER WITHIN MAP.	13	6-9(4)
SECOND NODE MAP NUMBER.	4924	10-14(5)
SECOND NODE NUMBER WITHIN MAP.	17	15-18(4)
LINK DISTANCE IN TENTHS OF A KILOMETER BETWEEN FIRST AND SECOND NODES.	18.7	19-23(5)
NUMBER OF EQUIVALENT LANES.	3	24-26(3)
ROAD TYPE CODE.	3	27-28(2)
GROUND COVER CODE.	3	29-30(2)
TERRAIN CODE.	2	31-32(2)
BRIDGE CODE.	0	33-34(2)
CITY CODE.	1	35-36(2)

CREATE MAP+DIRECTORY ENTRY FOR THIS MAP

REF
PAGE

```
.AT THIS POINT IN THE PROGRAM, THIS MAP'S NODE+VECTORS HAVE
.BEEN READ INTO CORE, AND ALL PERTINENT INFORMATION HAS BEEN
.PLACED IN THE PROPER AREA. THE MAP+NUMBER IS IN THE FIRST
.WORD OF THE NODE+VECTOR ARRAY, AND THE NUMBER OF NODES IN
.THIS MAP IS KNOWN FROM A COUNTER WHICH WAS INCREMENTED EVERY TIME
.A NEW NODE CARD WAS READ.
.
MAP+DIRECTORY ENTRY(1,NUMBER OF MAPS)=MAP+NUMBER
MAP+DIRECTORY ENTRY(2,NUMBER+OF+MAPS)=NUMBER OF NODES
```

CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK

REF
PAGE

```
.BEFORE THESE THREE FILES MAY BE ESTABLISHED, THE DATA MUST BE
.PLACED ON COMPUTER CARDS. THIS DATA IS OBTAINED BY PLACING A
.93 X 88 GRID OVER THE APPROPRIATE MAP. EACH SQUARE OF THE GRID
.IS ASSIGNED A VALUE EQUAL TO THE CODE WHICH MOST DESCRIBES THE
.AREA. THAT IS, IF THE AREA UNDERNEATH THE SQUARE IS PARTIALLY
.CODE 4, AND PARTIALLY CODE 5, THE VALUE ASSIGNED IS EITHER 4 OR
.5, DEPENDING UPON WHICH APPEARS TO OCCUPY THE MAJORITY OF THE
.SQUARE. IF MORE THAN 2 CODES ARE COVERED, THEN THE CODE CHOSEN
.IS THE ONE WHICH DOMINATES (SEEMS TO BE GREATER THAN THE OTHERS
.TAKEN INDIVIDUALLY) THE SQUARE.
.
do FOR EACH MAP
    CREATE DATA CARDS
enddo FOR EACH MAP
.
.WITH THE CARDS ALL MADE, IT IS NOW A MATTER OF READING THE CARDS
.INTO MEMORY, CONVERTING THE VALUES AS APPROPRIATE, MAKING A DIR-
.ECTORY ENTRY, AND WRITING THE RESULTS OUT TO DISK.
.
do ONCE
    CREATE DISK FILE FOR CROSS-COUNTRY MAPS, 'CCM.DAT'
    CREATE DISK FILE FOR WINTER CONCEALMENT MAPS, 'WCON.DAT'
    CREATE DISK FILE FOR SUMMER CONCEALMENT MAPS, 'SCON.DAT'
    CREATE DISK FILE FOR DIRECTORY, 'CCMDIR.DAT'
enddo ONCE
.
.BECAUSE THE ALLOWABLE CODES HAVE THE VALUES 1-7 FOR CROSS-COUN-
.TRY AND 8-11 FOR CONCEALMENT, ONLY FOUR BITS ARE NEEDED TO STORE
.THIS INFORMATION. A GOOD SPACE SAVING MAY BE MADE IF THE VALUES
.ARE PACKED FOUR TO A WORD. DOING THIS, THOUGH, MEANS THAT, IN
.ORDER TO DEAL WITH COMPLETE ROWS OF DATA, FOUR ROW OF CARDS
.SHOULD BE PROCESSED AT ONE TIME.
```

```
*      .                                                                          *
*     do FOR EACH MAP                                                             *
*        do UNTIL ALL ROWS ARE READ IN                                            *
*           .                                                                     *
*           .SINCE EACH ROW HAS 93 ELEMENTS, AND A COMPUTER CARD HAS ONLY         *
*           .72 COLUMNS, EACH ROW USES TWO CARDS.                                 *
*           .                                                                     *
*           READ FOUR ROWS WORTH OF CARDS (EIGHT ALTOGETHER)                      *
*           PACK THESE ROWS FOUR VALUES TO A WORD                                 *
*        enddo UNTIL ALL ROWS ARE READ IN                                         *
*        .                                                                        *
*        .ALL DATA IS NOW IN MEMORY. WRITE IT OUT TO PROPER DATA FILE.            *
*        .                                                                        *
*        WRITE DATA TO DISK                                                       *
*        .                                                                        *
*        .MAKE DIRECTORY ENTRY                                                    *
*        .                                                                        *
*        PROMPT FOR MAP+NUMBER                                                    *
*        READ INPUT                                                               *
*        PLACE INPUT AT NEXT AVAILABLE PLACE IN CCM.DIR ...ONLY DO THIS IF CROSS-COUNTRY MAP   /*
*              BEING PROCESSED.                                                   *
*        PROMPT FOR UTM LATITUDE AND LONGITUDE EXTREMES                           *
*        PLACE INPUT AT NEXT AVAILABLE PLACE IN CCM.DIR ...SAME ADMONITION AS BEFORE.  *
*        .                                                                        *
*        .WRITE DIRECTORY TO DISK                                                 *
*        .                                                                        *
*        WRITE DIRECTORY TO DISK                                                  *
*     enddo FOR EACH MAP                                                          *
*                                                                                 *
***********************************************************************************
```

```
**************
*            *
* LAMAS FLOW *
*            *
**************
```

LAMAS

```
REF
PAGE
        do UPON MAIN+PROGRAM EXECUTION
            do FOREVER
                DISPLAY MAIN FUNCTION MENU
                WAIT FOR USER INPUT
                do CASE OF :
     ROAD+NETWORK:
 36                 IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE+VECTOR DATA BASE
     TERRAIN+MODEL:
 66                 IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE
                    CONCEALMENT DATA BASES
     EXIT:
                    EXIT THIS SYSTEM
                enddo CASE OF
            enddo FOREVER
        enddo UPON MAIN+PROGRAM EXECUTION
```

IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE+VECTOR DATA BASE

```
REF
PAGE ****************************************************************
     *                                                              *
     *      do UPON FUNCTION REQUEST                                *
     *          do FOREVER                                          *
     *              DISPLAY FUNCTION MENU                           *
     *              WAIT FOR USER INPUT                             *
     *              do CASE OF                                      *
     *   INITIALIZE:                                                *
  37 *                  PREPARE FOR ROAD NETWORK PATH CALCULATIONS  *
     *   CALCULATION:                                               *
  45 *                  PATH DETERMINATION AND DISPLAY              *
     *   EXIT:                                                      *
     *                  return TO CALLING PROGRAM                   *
     *              enddo CASE OF                                   *
     *          enddo FOREVER                                       *
     *      enddo UPON FUNCTION REQUEST                             *
     *                                                              *
     ****************************************************************
```

PREPARE FOR ROAD NETWORK PATH CALCULATIONS

```
REF
PAGE
        do UPON FUNCTION REQUEST
            do FOREVER
                DISPLAY FUNCTION MENU
                WAIT FOR USER INPUT
                do CASE OF
     MAPS:
 38                 INITIALIZE MAP+NUMBERS AND SCREEN EXTREMES
     UNITS:
 42                 INITIALIZE UNIT+VECTORS
     FEBA:
 43                 ESTABLISH A FORWARD EDGE OF BATTLE AREA
     EXIT:
                    return TO CALLING PROGRAM
                enddo CASE OF
            enddo FOREVER
        enddo UPON FUNCTION REQUEST
```

INITIALIZE MAP←NUMBERS AND SCREEN EXTREMES

```
REF
PAGE
      do UPON FUNCTION REQUEST
39       ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS
         do FOR EACH NODE IN MEMORY
            do FOR EACH ADJACENT NODE
               if ADJACENT NODE'S MAP←NUMBER IS IN THE DIRECTORY
41                CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX
               endif ADJACENT NODE'S MAP←NUMBER
            enddo FOR EACH ADJACENT NODE
         enddo FOR EACH NODE IN MEMORY
         PROMPT FOR ENTRY OF SCREEN EXTREMES OF LATITUDE AND LONGITUDE
         READ INPUT
         PLACE INPUT IN COMMON STORAGE
         return TO CALLING PROGRAM
      enddo UPON FUNCTION REQUEST
```

ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS

REF
PAGE

```
. THIS DIRECTORY CONSISTS OF A NUMBER OF ENTRIES, ONE FOR EACH
.MAP WHICH THE USER WISHES TO EMPLOY. THE FIRST DIRECTORY ENTRY CORRE-
.SPONDS TO THE FIRST MAP←NUMBER REQUESTED BY THE USER, AND SO ON FOR
.EACH ENTRY. THERE ARE THREE ELEMENTS IN EACH ENTRY, 1) THE MAP←NUMBER,
.2) THE NUMBER OF NODES IN THE MAP, AND 3) THE NUMBER OF NODES ALREADY
.PRESENT IN MEMORY.
. GIVEN THIS MAP←DIRECTORY, IT IS A SIMPLE MATTER TO READ A MAP'S
.NODES INTO MAIN MEMORY, SINCE WE KNOW HOW MANY TO READ IN,
.MAP←DIRECTORY WORD 2, AND WHERE TO START PUTTING THEM, MAP←DIRECTORY
.WORD 3 +1.
.
READ DISK FILE MAP←DIRECTORY INTO MAIN MEMORY
PROMPT FOR ENTRY OF MAP←NUMBERS
READ USER INPUT OF MAP←NUMBERS TO BE PROCESSED
do FOR EACH MAP←NUMBER ENTERED BY USER
    if MAP←NUMBER IS IN THE DISK FILE MAP←DIRECTORY...AS THE I TH ENTRY
        SET FIRST WORD OF MAIN MEMORY MAP←DIRECTORY TO I
        SET SECOND WORD TO NUMBER OF NODES IN THIS MAP
        SET THIRD WORD TO NUMBER OF NODES ALREADY PRESENT IN MEMORY
40      READ THIS MAP'S NODES INTO MAIN MEMORY
        RESET FIRST WORD TO THIS MAP←NUMBER
    else MAP←NUMBER IS NOT IN THE DATA BASE
        PRINT ERROR MESSAGE
        LET THE USER RE-ENTER THE MAP←NUMBER
        cycle
    endif MAP←NUMBER IS IN THE DISK FILE MAP←DIRECTORY
enddo FOR EACH MAP←NUMBER ENTERED BY USER
```

READ THIS MAP'S NODES INTO MAIN MEMORY

REF
PAGE

```
. AT THIS POINT IN THE PROGRAM, THE MAP←DIRECTORY IN MAIN MEMORY
.HAS ITS AS ELEMENTS 1) THE INDEX I, INDICATING THAT THIS MAP IS THE
.I TH ONE IN THE DISK FILE MAP←DIRECTORY, 2) NUMBER OF NODES IN THIS MAP
.AND 3) NUMBER OF NODES ALREADY PRESENT IN MEMORY. IN ORDER TO
.READ THIS MAP'S NODES INTO MEMORY SO THAT NO SPACE IS WASTED,
.WE NEED ONLY CONSIDER THE INFORMATION JUST LISTED. TO READ DATA
.FROM DISK TO MEMORY WE NEED TO KNOW HOW MANY BYTES WILL BE MOVED.
.THIS IS JUST WORD 2 MULTIPLIED BY 64 (32 WORDS * 2 BYTES PER WORD). WE
.ALSO MUST KNOW EXACTLY WHERE THE DATA WILL BE PLACED IN MEMORY, THAT
.IS, THE NEXT AVAILABLE SPACE IN THE NODE←VECTOR ARRAY.
.THIS IS SIMPLY WORD 3 PLUS ONE. FINALLY, WE MUST KNOW FROM WHAT
.DISK BLOCK IS THE DATA COMING. THIS IS I-1 MULTIPLIED BY 10, SINCE
.THE FIRST MAP STARTS AT BLOCK ONE, THE SECOND AT BLOCK 11, ETC.
.NOTE THAT NO MAP HAS MORE THAN 80 NODE←VECTORS, EACH VECTOR
.HAVING 32 WORDS, SO THERE ARE 8 NODE←VECTORS TO A DISK BLOCK OF
.256 WORDS.
.
BYTES = (MAP←DIRECTORY WORD 2) * 64
INDEX = (MAP←DIRECTORY WORD 3) + 1
STARTING BLOCK = ((MAP←DIRECTORY WORD 1) -1) * 10
PERFORM DISK READ (BYTES, INDEX, STARTING BLOCK)
```

CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX

REF
PAGE

```
*******************************************************************************
*
*  .IN ORDER TO ACCESS A NODE QUICKLY, IT IS NECESSARY THAT ONE NOT            *
*  .HAVE TO USE THE MAP←DIRECTORY EACH TIME A REFERENCE IS MADE.               *
*  .THUS, CONSIDER THE NODE←VECTOR ARRAY AS A LIST OF NODES,                   *
*  .INDEXABLE BY INTEGERS, SUCH THAT THE FIRST NODE←VECTOR HAS INDEX=1,        *
*  .THE SECOND HAS INDEX=2, AND SO ON FOR THE ENTIRE LIST. THEN, IF            *
*  .WE KNOW THE MAP←NUMBER AND THE NODE←NUMBER OF A PARTICULAR NODE←VECTOR,    *
*  .THEN THIS INDEX MAY BE CALCULATED ONCE, AND SAVED. THE MOST LOGICAL        *
*  .PLACE TO SAVE IT IS IN IT'S ADJACENT MAP←NUMBER LOCATION.                  *
*  . THIS CALCULATION IS SIMPLE, AND DEPENDS ON OUR KNOWLEDGE OF JUST          *
*  .TWO THINGS, 1) FOR THE ADJACENT NODE'S MAP←NUMBER, THE NUMBER OF NODES     *
*  .ALREADY IN CORE, AND 2) FOR THE ADJACENT NODE, ITS NODE←NUMBER. BOTH       *
*  .OF THESE PIECES OF INFORMATION ARE READILY AVAILABLE, THE FIRST IS JUST    *
*  .THE THIRD WORD OF THE ADJACENT NODE'S MAP←DIRECTORY ENTRY, AND             *
*  .THE SECOND IS IN THE NODE←VECTOR.                                          *
*  .                                                                           *
*  ADJACENT MAP←NUMBER = (ADJACENT NODE'S MAP←DIRECTORY WORD 3) + ADJACENT NODE←NUMBER
*                                                                              *
*******************************************************************************
```

INITIALIZE UNIT←VECTORS

REF
PAGE

```
. THIS DATA BASE IS COMPLETELY USER DEFINED AT EXECUTION TIME.
.THE USER SPECIFIES UNIT NAME, UNIT STARTING AND DESTINATION LO-
.CATIONS, UNIT PRIORITY, AND UNIT TYPE CODE.
.
do UPON FUNCTION REQUEST
    ASK IF USER WISHES TO REINITIALIZE OR ADD ON TO EXISTING STRUCTURE
    if USER WANTS TO REINITIALIZE
        ERASE ALL UNIT←VECTORS
        SET UNIT←COUNTER TO ZERO
    endif USER WANTS TO REINITIALIZE
    .
    .PERFORM THE FOLLOWING AS MANY AS SIXTY TIMES
    .
    do UNTIL UNIT←COUNTER EQUALS SIXTY
        .
        .PROMPT FOR THE FOLLOWING: UNIT NAME, STARTING LOCATION, DESTINATION,
        .PRIORITY, AND TYPE CODE. ERROR CHECKING IS BE PERFORMED ON ALL INPUT
        .SAVE THE UNIT NAME. INPUT FOR THE START AND FINISH LOCATIONS IS IN THE
        .FORM MAP←NUMBER, NODE←NUMBER. ANY TIME THERE IS AN ERROR, THE PROMPT
        .WILL BE REPEATED UNTIL VALID INPUT IS RECEIVED.
        .
        PROMPT USER TO ENTER UNIT NAME
        if UNIT NAME IS NULL
            return TO CALLING PROGRAM
        else INPUT IS VALID CHARACTERS
            INCREMENT UNIT←COUNTER
            ASSIGN INPUT TO FIRST FOUR WORDS OF NEXT AVAILABLE UNIT←VECTOR
            PROMPT FOR STARTING LOCATION
            if STARTING LOCATION IS INVALID
                PRINT ERROR MESSAGE
                TRY AGAIN
            else INPUT IS VALID
```

```
                    ASSIGN INPUT TO UNIT+VECTOR WORD 5 (STARTING LOCATION)
                    ASSIGN INPUT TO UNIT+VECTOR WORD 7 (PRESENT LOCATION)
                    PROMPT FOR DESTINATION
                    if DESTINATION IS INVALID
                        PRINT ERROR MESSAGE
                        TRY AGAIN
                    else INPUT IS VALID
                        ASSIGN INPUT TO UNIT+VECTOR WORD 6 (DESTINATION)
                        PROMPT FOR PRIORITY
                        if PRIORITY IS INVALID
                            PRINT ERROR MESSAGE
                            TRY AGAIN
                        else INPUT IS VALID
                            ASSIGN INPUT TO FIRST BYTE OF UNIT+VECTOR WORD 8
                            PROMPT FOR TYPE CODE
                            if TYPE CODE IS INVALID
                                PRINT ERROR MESSAGE
                                TRY AGAIN
                            else INPUT IS VALID
                                ASSIGN INPUT TO SECOND BYTE OF UNIT+VECTOR WORD 8
                            endif TYPE CODE IS INVALID
                        endif PRIORITY IS INVALID
                    endif DESTINATION IS INVALID
                endif STARTING LOCATION IS INVALID
            endif UNIT NAME IS NULL
        enddo UNTIL UNIT+COUNTER EQUALS SIXTY
    enddo UPON FUNCTION REQUEST
```

ESTABLISH A FORWARD EDGE OF BATTLE AREA

REF
PAGE

```
.AN ARBITRARY MAXIMUM OF SIX NODES MAY BE DESIGNATED AS FEBA POINTS.
.
do UPON FUNCTION REQUEST
    do SIX TIMES
        PROMPT FOR MAP←NUMBER AND NODE←NUMBER, OR EXIT
        if INPUT IS EXIT
            return TO CALLING PROGRAM
        endif INPUT IS EXIT
        CHECK MAP←NUMBER AND NODE←NUMBER FOR LEGITIMACY
        if NOT VALID
            PRINT ERROR MESSAGE
            TRY AGAIN
        else VALID
            FIND NODE←VECTOR CORRESPONDING TO MAP←NUMBER, NODE←NUMBER
            ASSIGN NODE←VECTOR TO FEBA ARRAY
        endif NOT VALID
    enddo SIX TIMES
enddo UPON FUNCTION REQUEST
```

FIND NODE←VECTOR NEAREST TO UTM COORDINATES

REF
PAGE

```
.THIS WILL BE ACCOMPLISHED BY STARTING AT THE BEGINNING OT THE
.NODE←VECTOR ARRAY AND COMPUTING EACH NODE←VECTOR 'S DISTANCE FROM THE
.UTM COORDINATES USING A LEAST SQUARES APPROACH. THAT IS, DISTANCE IS
.CALCULATED TO BE THE SUM OF THE SQUARES OF THE DIFFERENCES BETWEEN THE
.NODE←VECTOR 'S LATITUDE AND LONGITUDE AND THE CALCULATED UTM COORDI-
.NATES.
.
LEAST = INFINITY
do FOR EACH NODE←VECTOR
    DISTANCE = (UTM LAT. - LAT.) ** 2 + (UTM LON. - LON.) ** 2
    if DISTANCE IS LESS THAN LEAST
        REMEMBER THE NODE←VECTOR
        LEAST = DISTANCE
    endif DISTANCE IS LESS THAN LEAST
enddo FOR EACH NODE←VECTOR
.
.REMEMBERED NODE←VECTOR IS THE ONE WE WANT
.
```

PATH DETERMINATION AND DISPLAY

```
REF
PAGE ************************************************************************************
     *                                                                                  *
     *  .THESE ARE THE FUNCTIONS WHICH PERFORM THE FUNDAMENTAL MOVEMENT ALGO-           *
     *  .RITHM. THE MAIN PURPOSE IS TO COMPUTE OPTIMAL PATHS FOR DEFINED                *
     *  .GROUND FORCE UNITS, SO THAT CERTAIN CRITERIA IS SATISFIED. THE GENERAL         *
     *  .PROCEDURE IS, GIVEN A UNIT'S STARTING NODE, DESTINATION NODE, AND              *
     *  .DESIRED ARRIVAL TIME AT DESTINATION, OR START TIME AT THE BEGINNING,           *
     *  .CALCULATE THE BEST PATH FOR THE UNIT TO TRAVEL. 'BEST' IS USER-DEFINED         *
     *  .AS A FUNCTION OF TIME AND RISK. THE FORMULA USED IS K*TIME + C*RISK,           *
     *  .WHERE K AND C ARE CONSTANTS, C EQUALING ZERO OR ONE, AND K BEING BE-           *
     *  .TWEEN ZERO AND TEN. BOTH K AND C MAY NOT BE ZERO AT THE SAME TIME.             *
     *  .USING THE BASIC PATH ALGORITHM, CERTAIN VARIATIONS OF THE PATH CALCU-          *
     *  .LATION MAY BE PERFORMED, SUCH AS FINDING THE SECOND SHORTEST PATH,             *
     *  .FINDING THE OPTIMAL NODE FOR INTERDICTION, AND, GIVEN A NUMBER OF UNITS        *
     *  .TRAVELING THROUGH A ROAD NETWORK, FIND WHICH ORDER OF PRIORITIES OPTI-         *
     *  .MIZES THE ENTIRE NETWORKS USAGE.                                               *
     *  . THERE ARE TWO MORE OPTIONS FROM WHICH THE USER MAY CHOOSE, INTERDIC-          *
     *  .TION AND SOLUTIONS. THEIR PROPERTIES WILL BE EXPLAINED LATER.                  *
     *  .                                                                               *
     *  do UPON FUNCTION REQUEST                                                        *
     *      do FOREVER                                                                  *
     *          DISPLAY FUNCTION MENU                                                   *
     *          WAIT FOR USER INPUT                                                     *
     *          do CASE OF                                                              *
     * INTERDICTION:                                                                    *
  46 *              PERFORM INTERDICTIVE OPERATIONS                                     *
     * PATH:                                                                            *
  53 *              CALCULATE PATH(S) AND SOLUTION VECTORS                              *
     * SECOND:                                                                          *
  60 *              CALCULATE SECOND BEST PATH                                          *
     * NETWORK:                                                                         *
  61 *              FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL                *
     * BEST:                                                                            *
  62 *              CALCULATE BEST NODE AT WHICH TO INTERDICT                           *
```

```
   *  SOLUTION:                                                   *
63 *                PRESENT RESULTS                               *
   *  EXIT:                                                       *
   *                return TO CALLING PROGRAM                     *
   *            enddo CASE OF                                     *
   *        enddo FOREVER                                         *
   *    enddo UPON FUNCTION REQUEST                               *
   *                                                              *
   ****************************************************************
```

PERFORM INTERDICTIVE OPERATIONS

```
REF
PAGE *****************************************************************************
     *                                                                           *
     *   . HERE, THE USER MAY INTERDICT TO FORCE UNITS TO MOVE A CERTAIN         *
     *   .WAY, TO MAKE LINKS IMPASSIBLE, OR TO SEE THE EFFECTS OF SUCH INTERDIC- *
     *   .TION. THE USER MAY ALSO LOOK AT THE CONTENTS OF NODE<VECTORS.          *
     *   .                                                                       *
     *   do UPON FUNCTION REQUEST                                                *
     *       do FOREVER                                                          *
     *           DISPLAY FUNCTION MENU                                           *
     *           WAIT FOR USER INPUT                                             *
     *           do CASE OF                                                      *
     * VECTOR2:                                                                  *
  47 *               PRINT MAP<NUMBER, NODE<NUMBER OF NODE<VECTOR NEAREST GIVEN COORDINATES *
     * CONTENTS:                                                                 *
  48 *               CHANGE CONTENTS OF A NODE<VECTOR                            *
     * ERASE:                                                                    *
  50 *               DELETE A NODE FROM THE NODE<VECTOR ARRAY                    *
     * SHOW:                                                                     *
  51 *               PRINT CONTENTS OF A NODE<VECTOR                             *
     * PROPERTIES:                                                               *
  52 *               CHANGE PROPERTIES OF A UNIT                                 *
     * EXIT:                                                                     *
     *               return TO CALLING PROGRAM                                   *
     *           enddo CASE OF                                                   *
     *       enddo FOREVER                                                       *
     *   enddo UPON FUNCTION REQUEST                                             *
     *                                                                           *
     *****************************************************************************
```

PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINATES

REF
PAGE

```
. THIS ALLOWS THE USER TO FIND OUT WHICH NODE←VECTOR IS NEAREST ANY UTM
.COORDINATES OF INTEREST. THIS IS OFTEN HELPFUL IF THE TRACKBALL IS
.UNAVAILABLE.
.
PROMPT FOR USER INPUT UTM COORDINATES
SET LEAST TO INFINITY
do FOR EACH NODE←VECTOR
    DISTANCE = (UTM LAT.-LATITUDE) ** 2 + (UTM LON. - LONGITUDE) **2
    if DISTANCE IS LESS THAN LEAST
        SET LEAST EQUAL TO DISTANCE
        REMEMBER NODE←VECTOR
    endif DISTANCE IS LESS THAN LEAST
enddo FOR EACH NODE←VECTOR
.
.THE ANSWER IS THE LAST REMEMBERED NODE←VECTOR
.
PRINT OUT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR
```

CHANGE CONTENTS OF A NODE+VECTOR

```
REF
PAGE ****************************************************************************
     *                                                                          *
     *  . THIS FUNCTION ALLOWS THE USER TO CHANGE CERTAIN CONDITIONS OF ANY     *
     *  .NODE+VECTOR. THE USER MAY CHANGE ANY OF SIX LINK CHARACTERISTICS,      *
     *  .NUMBER OF LANES, LINK DISTANCE, TERRAIN CODE, BRIDGE CODE, CITY CODE,  *
     *  .OR ROAD TYPE.                                                          *
     *  .                                                                       *
     *  do UPON FUNCTION REQUEST                                                *
     *      PROMPT ENTRY OF MAP+NUMBER, NODE+NUMBER                             *
     *      if ENTERED NODE+VECTOR IS NOT IN MEMORY                             *
     *          PRINT ERROR MESSAGE                                             *
     *          TRY AGAIN                                                       *
     *      else ENTERED NODE+VECTOR IS IN MEMORY                               *
     *          PROMPT FOR LINKNG MAP+NUMBER, NODE+NUMBER                       *
     *          if LINKING NODE+VECTOR IS NOT IN MEMORY                         *
     *              PRINT ERROR MESSAGE                                         *
     *              TRY AGAIN                                                   *
     *          else LINKING NODE+VECTOR IS IN MEMORY                           *
     *              if LINKING NODE+VECTOR DOES NOT LINK TO FIRST ENTERED NODE+VECTOR *
     *                  PRINT ERROR MESSAGE                                     *
     *                  TRY AGAIN                                               *
     *              else TWO NAMED NODE+VECTORS ARE IN MEMORY AND ARE LINKED    *
     *                  PRINT LIST OF CHOICES FOR ALTERATION                    *
     *                  WAIT FOR USER INPUT                                     *
     *                  do CASE OF                                              *
     * ROAD:                                                                    *
  49 *                      CHANGE CHARACTERISTIC ...ROAD TYPE                  *
     * LINK:                                                                    *
  49 *                      CHANGE CHARACTERISTIC ...LINK DISTANCE              *
     * TERRAIN:                                                                 *
  49 *                      CHANGE CHARACTERISTIC ...TERRAIN CODE               *
     * LANES:                                                                   *
  49 *                      CHANGE CHARACTERISTIC ...NUMBER OF LANES            *
     * BRIDGE:                                                                  *
```

```
49 * 33                    CHANGE CHARACTERISTIC ...BRIDGE CODE                                 *     1171
   * 34 CITY:                                                                                   *     1172
49 * 35                    CHANGE CHARACTERISTIC ...CITY CODE                                   *     1173
   * 36                 enddo CASE OF                                                           *     1174
   * 37                 return TO CALLING PROGRAM                                               *     1175
   * 38              endif LINKING NODE<VECTOR DOES NOT LINK TO FIRST ENTERED NODE<VECTOR       *     1176
   * 39           endif LINKING NODE<VECTOR IS NOT IN MEMORY                                    *     1177
   * 40        endif ENTERED NODE<VECTOR IS NOT IN MEMORY                                       *     1178
   * 41     enddo UPON FUNCTION REQUEST                                                         *     1179
   *                                                                                            *
   **********************************************************************************************
```

CHANGE CHARACTERISTIC

REF
PAGE

```
. THIS IS THE SAME PROCEDURE REGARDLESS OF WHICH CHARACTERISTIC IS BEING
.CHANGED.
.
PRINT OUT OLD VALUE
PROMPT FOR NEW VALUE
if NEW VALUE WITHIN ACCEPTABLE LIMITS
    ASSIGN NEW VALUE TO NODE+VECTOR
else NEW VALUE IS NOT ACCEPTABLE
    PRINT ERROR MESSAGE
    TRY AGAIN
endif NEW VALUE IS WITHIN ACCEPTABLE LIMITS
```

DELETE A NODE FROM THE NODE←VECTOR ARRAY

REF
PAGE

```
. FOR THE PURPOSE OF INTERDICTION IT MAY BE DESIRED THAT A NODE←VECTOR
.BE 'ERASED', THAT IS, AFFECTED SO THAT NO UNIT WILL TRAVEL THROUGH IT.
.THIS ROUTINE CHANGES THE NUMBER OF LANES PARAMETER FOR EACH ADJACENT
.LINK TO 0. IF THERE'S NO ROAD, NO UNIT CAN TRAVEL THERE. ONCE
.DONE, RESTORATION IS PERFORMED BY EITHER REESTABLISHING EACH LINK
.MANUALLY, OR BY READING THE MAPS FROM DISK AGAIN.
.
do UPON FUNCTION REQUEST
    PROMPT FOR MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR TO BE ERASED
    if ENTERED VALUES ARE INVALID
        PRINT ERROR MESSAGE
        TRY AGAIN
    else ENTERED VALUES ARE VALID
        do FOR EACH ADJACENT NODE
            SET NUMBER OF LANES TO THIS NODE TO ZERO
        enddo FOR EACH ADJACENT NODE
        return TO CALLING PROGRAM
    endif ENTERED VALUES ARE INVALID
enddo UPON FUNCTION REQUEST
```

PRINT CONTENTS OF A NODE←VECTOR

REF
PAGE

```
do UPON FUNCTION REQUEST
    PROMPT FOR MAP←NUMBER, NODE←NUMBER
    if ENTERED VALUES ARE NOT VALID
        PRINT ERROR MESSAGE
        TRY AGAIN
    else ENTERED VALUES ARE VALID
        PRINT CONTENTS OF NODE←VECTOR
        return TO CALLING PROGRAM
    endif ENTERED VALUES ARE NOT VALID
enddo UPON FUNCTION REQUEST
```

CHANGE PROPERTIES OF A UNIT

REF
PAGE

```
. AT THIS TIME, THE ONLY PROPERTY WHICH WE WILL ALLOW TO BE CHANGED,
.IS THE UNIT'S PRIORITY.
.
do UPON FUNCTION REQUEST
    PROMPT ENTRY OF UNIT NAME
    if INPUT NAME HAS AN ENTRY IN UNITS FILE
        PRINT OLD VALUE
        PROMPT ENTRY OF NEW PRIORITY
        if INPUT IS BETWEEN 0 AND 127
            SET NEW VALUE IN UNIT←VECTOR
            return TO CALLING PROGRAM
        else INPUT IS NOT VALID
            PRINT ERROR MESSAGE
            TRY AGAIN
        endif INPUT IS BETWEEN 0 AND 127
    else INPUT NAME HAS NO ENTRY IN UNITS FILE
        PRINT ERROR MESSAGE
        TRY AGAIN
    endif INPUT NAME HAS AN ENTRY IN UNITS FILE
enddo UPON FUNCTION REQUEST
```

CALCULATE PATH(S) AND SOLUTION VECTORS

```
REF
PAGE
      . THIS ROUTINE WILL DETERMINE THE BEST PATH FOR GROUND FORCE
      .UNITS TO USE, TAKING INTO CONSIDERATION TRAVEL TIME, GROUND COVER, TER-
      .RAIN, BRIDGES, AND CITIES.
      . BEST DOESN'T ONLY IMPLY FASTEST, BUT MAY MEAN LEAST RISKY, OR SOME
      .COMBINATION OF SPEED AND RISK.
      .THERE ARE SEVERAL OPTIONS FROM WHICH THE USER IS FREE TO CHOOSE. AMONG
      .THESE ARE:
      . 1) SIMPLE BEST PATH CALCULATION, REGARDLESS OF TIME CONFLICTS,
      . 2) AUTOMATIC CONFLICT RESOLUTION; THE PROGRAM DOES THE RESOLUTION
      .WITHOUT USER INTERVENTION.
      . 3) WORK FORWARDS IN TIME, THAT IS, GIVE A STARTING TIME AND NODE,
      .AND A DESTINATION NODE. THE PROGRAM WILL THEN WORK FORWARD IN TIME,
      .BEGINNING AT THE START NODE, AND ENDING AT THE DESTINATION NODE.
      . 4) WORK BACKWARDS IN TIME. GIVEN A STARTING NODE, DESTINATION NODE
      .AND ARRIVAL TIME, START AT THE DESTINATION NODE, WORK BACKWARDS IN TIME
      .TO THE STARTING NODE.
      . AFTER THE PATH HAS BEEN DETERMINED, THE ROUTE IS RECONSTRUCTED USING
      .SOLUTION-VECTORS. THESE VECTORS RECORD THE TIMES DURING WHICH THE VAR-
      .IOUS NODES ARE BUSY. THIS INFORMATION WILL BE USED WHEN RESOLVING
      .TIME CONFLICTS AMONG THE PATHS.
      .
      do UPON FUNCTION REQUEST
          .
          .BEFORE ANYTHING ELSE, CLEAR AWAY ALL OLD ROUTES AND SOLUTIONS
          .
54        PURGE ALL ROUTES
          .
          .NOW GET PATH PARAMETERS
          .
          PROMPT FOR ENTRY OF MOVEMENT TYPE (BACKWARD OR FORWARD IN TIME)
          PROMPT FOR ENTRY OF TIME CONFLICT RESOLUTION CODE (NONE OR AUTOMATIC)
          .
```

```
*    .HAVE USER ERTER C AND K, THE COEFFICIENTS FOR RISK AND TIME          *
*    .                                                                      *
*    PROMPT FOR ENTRY OF C                                                  *
*    if C NOT EQUAL TO 0 OR 1                                               *
*        PRINT MESSAGE                                                      *
*        TRY AGAIN                                                          *
*    endif C NOT EQUAL TO 0 OR 1                                            *
*    PROMPT FOR ENTRY OF K                                                  *
*    if K NOT BETWEEN 0 AND 9.99                                            *
*        PRINT ERROR MESSAGE                                                *
*        TRY AGAIN                                                          *
*    endif K NOT BETWEEN 0 AND 9.99                                         *
*    if C AND K EQUAL 0                                                     *
*        PRINT ERROR MESSAGE                                                *
*        TRY AGAIN                                                          *
*    endif C AND K EQUAL 0                                                  *
*    .                                                                      *
*    .PROMPT FOR UNIT NAMES TO BE CONSIDERED                                *
*    .                                                                      *
*    do UNTIL USER ENTERS NULL                                              *
*        PROMPT FOR ENTRY OF UNIT NAME                                      *
*        if INPUT IS NULL                                                   *
*            undo UNTIL USER ENTERS NULL                                    *
*        else INPUT IS NOT NULL                                             *
*            REMEMBER UNIT                                                  *
*        endif INPUT IS NULL                                                *
*    enddo UNTIL USER ENTERS NULL                                           *
*    .                                                                      *
*    .CALCULATE THE PATHS                                                   *
*    .                                                                      *
*    do FOR EACH ENTERED UNIT, IN ORDER OF PRIORITY                         *
*        if MOVEMENT TYPE IS BACKWARDS                                      *
*            PROMPT FOR ARRIVAL TIME                                        *
*        else MOVEMENT TYPE IS FORWARDS                                     *
*            PROMPT FOR LEAVING TIME                                        *
*        endif MOVEMENT TYPE IS BACKWARDS                                   *
*        .                                                                  *
*        .THE 'WORKING LIST' IS A FICTITIOUS ARRAY WHICH KEEPS TRACK OF VARIOUS *
```

```
      .QUANTITIES NECESSARY FOR THE PATH ALGORITHM. THESE ARE WORTH MEASURE,
      .PREDECESSOR NODE NUMBER, CUMMULATIVE TIME, CUMMULATIVE WORTH MEASURE,
      .TIME MEASURE, SOLUTION+VECTOR POINTER, LANE USED, RISK MEASURE, AND
      .PARK TIME. THESE VALUES ARE HELD IN THE NODE+VECTOR FOR EASY REFER-
      .ENCE.
      .
      if TIME CONFLICT RESOLUTION IS AUTOMATIC
55        REINITIALIZE 'WORKING LIST' NODE+VECTOR ENTRIES
          SET K TO FIRST NODE TO BE CONSIDERED (DEPENDENT ON MOVEMENT TYPE)
      else TIME CONLICT RESOLUTION IS NOT AUTOMATIC
          if THIS UNIT'S MOVEMENT TYPE, ENTERED TIME, AND UNIT TYPE DIFFERS FROM/
              THE PREVIOUS UNIT'S
55            REINITIALIZE 'WORKING LIST' NODE+VECTOR ENTRIES
              SET K TO FIRST NODE TO BE CONSIDERED (DEPENDENT ON MOVEMENT TYPE)
          endif THIS UNIT'S PARAMETERS DIFFER
      endif TIME CONFLICT RESOLUTION IS AUTOMATIC
      .
      .DURING THIS ALGORITHM, K IS THE NODE FROM WHICH THE ALGORITHM STARTS.
      .THIS IS DETERMINED BY THE MOVEMENT TYPE TO BE EITHER THE START NODE
      .(TYPE IS FORWARD) OR THE DESTINATION NODE (TYPE IS BACKWARD). I IS THE
      .FINAL NODE, THE OPPOSITE OF K (IF K IS THE START NODE, I IS THE DESTI-
      .NATION, AND VICE VERSA). J IS THE NODE UNDER CURRENT CONSIDERATION BY
      .THE ALGORITHM.
      .
      SET I TO END NODE (DEPENDENT UPON MOVEMENT TYPE)
      SET J TO K
      do UNTIL NODE I IS LABELED ... TEST IMMEDIATELY FOR EARLY EXIT
          if I IS LABELED
              undo UNTIL NODE I IS LABELED
          endif I IS LABELED
          if J ISN'T LABELED
              LABEL NODE J
              SET J'S CUMMULATIVE TIME
              do FOR EACH OF J'S ADJACENT NODES
                  if IT IS LABELED
                      cycle FOR EACH OF J'S ADJACENT NODES
                  else IT IS NOT LABELED
56                    CALCULATE AND ASSIGN TIME AND WORTH VALUES
```

```
      * 09               endif IT IS LABELED                                 *      1353
      * 10             enddo FOR EACH OF J'S ADJACENT NODES                  *      1354
      * 11           endif J ISN'T LABELED                                   *      1355
   57 * 12           COMPUTE NEXT NODE TO LABEL ...DEPENDENT UPON TYPE       *      1356
      * 13           SET J TO NEW NODE                                       *      1357
      * 14         enddo UNTIL NODE I LABELED                                *      1358
   59 * 15         COMPUTE SOLUTION<VECTORS AND ROUTE<VECTOR                 *      1359
      * 16       enddo FOR EACH UNIT                                         *      1360
      * 17       return TO CALLING PROGRAM                                   *      1361
      * 18     enddo UPON FUNCTION REQUEST                                   *      1362
      *                                                                      *
      ************************************************************************
****** TOO MANY LINES IN SEGMENT
```

PURGE ALL ROUTES

REF PAGE

```
. THIS ROUTINE WILL GET RID OF ALL PREVIOUS PATHS WHICH HAVE BEEN
.CALCULATED. THIS CONSISTS OF ERASING ALL SOLUTION←VECTORS AND
.ROUTE←VECTORS, RESETING THEIR COUNTERS TO ZERO, AND ERASING ALL POINT-
.ERS WHICH REFERENCE THESE ARRAYS.
.
do FOR EACH ROUTE←VECTOR
    SET CONTENTS TO 0
enddo FOR EACH ROUTE←VECTOR
do FOR EACH SOLUTION←VECTOR
    SET CONTENTS TO 0
enddo FOR EACH SOLUTION←VECTOR
SET COUNTERS TO 0
do FOR EACH NODE←VECTOR
    SET SOLUTION←VECTOR POINTER TO 0
enddo FOR EACH NODE←VECTOR
```

REINITIALIZE 'WORKING LIST' NODE+VECTOR ENTRIES

REF
PAGE

```
. THIS FUNCTION MUST BE PERFORMED EACH TIME WE WISH TO
.'WIPE THE SLATE CLEAN' AND START A NEW PATH-FINDING PROCEDURE.
.
do FOR EACH NODE+VECTOR
    SET TIME MEASURE TO ZERO
    SET WORTH MEASURE TO ZERO
    SET CUMMULATIVE TIME TO ZERO
    SET CUMMULATIVE WORTH TO 32767
    SET LABEL TO ZERO
    SET PREDECESSOR TO -1
    SET LANE TO ONE
    SET PARKING TIME TO ZERO
    SET RISK MEASURE TO 0
enddo FOR EACH NODE+VECTOR
```

CALCULATE AND ASSIGN TIME AND WORTH VALUES

REF
PAGE

```
. WHEN A NODE IS FOUND BY THE PATH ALGORITHM TO BE THE NEXT TO BE
.LABELED, THE TIME AND WORTH VALUES OF ALL OF ITS LINKS TO UNLABELED
.ADJACENT NODES ARE CALCULATED AND ASSIGNED. THAT PROCESS OF CALCULAT-
.ING AND ASSIGNING IS DONE BY THIS ROUTINE.
. THERE ARE TWO TYPES OF TIME AND WORTH: MEASURE AND CUMMULATIVE. TIME
.MEASURE AND WORTH MEASURE REFER TO THE VALUE CALCULATED FOR THE LINK
.BETWEEN A NODE AND ONE OF ITS UNLABELED ADJACENT NODES. CUMMULATIVE
.TIME AND CUMMULATIVE WORTH REFER TO THE ACCUMULATED TIME AND WORTH
.WHICH HAS BUILT UP DURING THE PATH CALCULATION. THUS CUMMULATIVE TIME
.IS THE TIME OF DAY AT WHICH A UNIT ARRIVES AT THE ADJACENT NODE, AND
.CUMMULATIVE WORTH IS THE WORTH OF THE PATH AT THAT POINT IN TIME.
. WORTH IS A COMBINATION OF RISK AND TIME, USING PARAMETERS SPECIFIED
.BY THE USER. THE EQUATION IS C * RISK + K * TIME, WHERE C MAY BE
.0 OR 1, AND K MAY BE BETWEEN 0 AND 9.99, BUT BOTH MAY NOT EQUAL 0
.AT THE SAME TIME. A BEST PATH IS A PATH WHICH MINIMIZES WORTH. BY
.ADJUSTING THE PARAMETERS C AND K, THE USER MAKES HIS OWN DEFINITION
.OF BEST. IT COULD BE THAT BEST IS THE LEAST RISKY (C=1,K=0), FASTEST
.(C=0,K=1), OR A COMBINATION OF THE TWO.
. RISK IS CALCULATED BY SUMMING THE NUMERICAL VALUES ASSIGNED TO THE
.FOUR RISK FACTORS: GROUND COVER, TERRAIN, BRIDGE, AND CITY CODES.
.
CALCULATE LINK TRAVERSAL TIME
if MOVEMENT TYPE IS FORWARD
    CUMMULATIVE TIME IS PREDECESSOR'S CUMMULATIVE TIME + LINK TIME
else MOVEMENT TYPE IS BACKWARD
    CUMMULATIVE TIME IS THIS NODE'S CUMMULATIVE TIME - LINK TIME
endif MOVEMENT TIME IS FORWARD
.
.RESOLVE TIME CONFLICTS
.
if USER DOES NOT WANT TIME CONFLICTS RESOLVED
    CONTINUE
```

```
elseif THIS NODE+VECTOR HAS NEVER BEEN USED BY A PATH
    CONTINUE
else THIS NODE+VECTOR HAS BEEN USED BY A PATH
    .
    .LOOK FOR TIME CONFLICT. IF ONE IS FOUND, RESOLVE IT
    .
    SET ALL FOUR LANES TO CUMMULATIVE TIME
    do FOR EACH SOLUTION+VECTOR REFERING TO THIS NODE
        if THE TIME INTERVAL SHOULD NOT BE CONSIDERED
            cycle FOR EACH SOLUTION+VECTOR
        else THE TIME INTERVAL SHOULD BE CONSIDERED
            SET INDEX TO SOLUTION+VECTOR 'S LANE NUMBER
            if THIS UNIT CAN CLEAR THIS NODE BEFORE THE TIME INTERVAL
                cycle FOR EACH SOLUTION+VECTOR
            else THIS UNIT CANNOT CLEAR THIS NODE BEFORE THE TIME INTERVAL
                if THIS UNIT DOES NOT ARRIVE UNTIL AFTER THE TIME INTERVAL
                    cycle FOR EACH SOLUTION+VECTOR
                else THIS UNIT ARRIVES DURING THE TIME INTERVAL
                    .
                    .THERE IS A TIME CONFLICT
                    .
                    if MOVEMENT TYPE IS FORWARD
                        HOLD UP UNIT UNTIL NODE CLEARS
                        RECORD THIS TIME IN LANE (INDEX)
                    else MOVEMENT TYPE IS BACKWARD
                        HAVE UNIT ARRIVE BEFORE TIME INTERVAL
                        RECORD THIS TIME IN LANE (INDEX)
                    endif MOVEMENT TYPE IS FORWARD
                    .
                    .THAT'S ALL FOR THIS SOLUTION+VECTOR
                    .
                endif THIS UNIT DOES NOT ARRIVE UNTIL AFTER THE TIME INTERVAL
            endif THIS UNIT CAN CLEAR THIS NODE BEFORE THE TIME INTERVAL
        endif THE TIME INTERVAL SOULD NOT BE CONSIDERED
    enddo FOR EACH SOLUTION+VECTOR REFERING TO THIS NODE
    .
    .FIND THE BEST LANE (FOR FORWARD, BEST IS LEAST TIME, FOR BACKWARD, BEST
    .IS THE GREATEST TIME).
```

```
    .
    if MOVEMENT IS FORWARD
        SET BEST TO 32767
    else MOVEMENT IS BACKWARD
        SET BEST TO 0
    endif MOVEMENT IS FORWARD
    do FOR EACH AVAILABLE LANE FOR THIS LINK
        if TIME OF LANE IS LESS (GREATER) THAN BEST
            SET BEST TO TIME OF LANE
            REMEMBER LANE NUMBER
        endif TIME OF LANE IS LESS (GREATER) THAN BEST
    enddo FOR EACH AVAILABLE LANE FOR THIS LINK
    .
    .ANY TIME CONFLICT HAS NOW BEEN RESOLVED. THE BEST RESULT FROM THE
    .PREVIOUS SECTION IS NOW THE 'TIME OF RECORD'.
    .
    CALCUATE PARKING TIME ('TIME OF RECORD' LESS CUMMULATIVE TIME)
    .
    .PARKING TIME AFFECTS THE ENTIRE LENGTH OF THE UNIT, SO IT MUST BE
    .SEEN IF THE ADITION OF THIS PARKING TIME CREATES ANY NEW CONFLICTS.
    .IF SO, THIS LINK CANNOT BE USED.
    .
    if PARKING TIME EQUALS ZERO
        CONTINUE
    else PARKING TIME DOES NOT EQUAL ZERO
        if PARKING TIME CREATES NEW CONFLICTS
            MAKE NO ASSIGNMENTS OF TIME AND WORTH
            return
        endif PARKING TIME CREATES NEW CONFLICTS
    endif PARKING TIME EQUALS ZERO
endif THE USER DOES NOT WANT TIME CONFLICTS RESOLVED
.
.NOW CALCULATE RISK AND MAKE ASSIGNMENTS
.
CALCULATE RISK
CALCULATE WORTH FOR THIS LINK
CALCULATE CUMMULATIVE WORTH
if CUMMULATIVE WORTH IS LESS THAN CURRENT CUMMULATIVE WORTH
```

```
* 09        MAKE ASSIGNMENTS OF TIME, WORTH, PARKING, AND LANE                    *     1503
* 10    endif CUMMULATIVE WORTH IS LESS THAN CURRENT CUMMULATIVE WORTH            *     1504
*                                                                                 *
***********************************************************************************
****** TOO MANY LINES IN SEGMENT
```

COMPUTE NEXT NODE TO LABEL

REF
PAGE

```
.
. THIS CONSISTS OF FINDING WHICH UNLABELED NODE VECTOR HAS THE
.LEAST CUMMULATIVE WORTH. BY MAKING WORTH A FUNCTION SUCH THAT
.THE BETTER A NODE'S WORTH,THE SMALLER THE WORTH VALUE, CUM-
.MULATIVE WORTH MUST BE ORIGINALLY SET TO INFINITY, OR IN THIS
.CASE, 32767.
.
SET LEAST TO 32767
do FOR EACH NODE
    if NODE IS LABELED
        cycle
    elseif WORTH MEASURE < LEAST
        SET LEAST TO WORTH MEASURE
        SAVE NODE NUMBER
    endif NODE IS LABELED
enddo FOR EACH NODE
```

EXPLANATION OF PATH DETERMINATION ALGORITHM

OUR PROBLEM IS TO DETERMINE A ROUTE FOR A GROUND FORCE UNIT TO FOLLOW, SUCH THAT THE PATH GETS THE UNIT TO ITS DESTINATION NODE FROM ITS STARTING NODE AS QUICKLY AND/OR AS LEAST RISKY AS POSSIBLE. THIS IS A SHORTEST PATH PROBLEM, WITH SOME TWISTS THROWN IN. ONE OF THESE TWISTS IS THAT EACH UNIT HAS A DIFFERENT RATE OF TRAVEL, WITH THAT RATE VARYING ACCORDING TO THE TIME OF DAY. ANOTHER IS THAT WE REALLY HAVE MORE THAN ONE UNIT, AND WHEN MORE THAN ONE UNIT HAS THE SAME DESTINATION NODE, WE MAY WISH FOR THE UNITS TO ARRIVE THERE AT THE SAME TIME. WE HAVE FOUND THAT THE SIMULTANEOUS DESTINATION TIME PROBLEM MAY BE SOLVED BY STARTING AT THE DESTINATION NODE AND WORKING OUR WAY BACK TO THE STARTING NODE, RUNNING TIME IN REVERSE.

PERHAPS IT WOULD BE BEST TO FIRST EXPLAIN DIJKSTRA'S ALGORITHM FOR ONE UNIT SUPPOSE THAT THERE ARE N NODES, NUMBERED FROM 1 TO N. AND SUPPOSE FURTHER THAT WE WISH TO FIND THE SHORTEST PATH BETWEEN NODES 1 AND N. THAT IS, NODE 1 IS THE DESTINATION, AND NODE N IS THE STARTING NODE. LABEL 1 WITH THE PERMANENT VALUE ZERO, AND TENTATIVELY LABEL ALL OTHERS WITH VALUE INFINITY. EACH TIME A PERMANENT LABEL IS ASSIGNED, IT REPRESENTS THE SHORTEST DISTANCE BETWEEN THAT NODE AND THE DESTINATION NODE, NODE N.

ONE BY ONE, COMPARE EACH NODE LABEL EXCEPT THAT AT 1 WITH THE SUM OF THE LABEL OF NODE 1 (THAT IS, 0) AND THE DIRECT DISTANCE FROM NODE 1 TO THE NODE IN QUESTION. THE SMALLER OF THE TWO NUMBERS IS THE NEW TENTATIVE LABEL.

NEXT, DETERMINE THE SMALLEST OF THE N-1 TENTATIVE LABELS AND DECLARE IT PERMANENT. SUPPOSE THAT NODE K IS THE ONE PERMANENTLY LABELED. THEN, ONE AT A TIME, COMPARE EACH OF THE N-2 REMAINING TENTATIVE NODE LABELS TO THE SUM OF THE LABEL JUST ASSIGNED PERMANENTLY TO NODE K AND THE DIRECT DISTANCE FROM NODE K TO THE NODE UNDER CONSIDERATION. THE SMALLER OF THE TWO NUMBERS BECOMES THE TENTATIVE LABEL. DETERMINE THE MINIMUM OF THE N-2 TENTATIVE LABELS, DECLARE IT PERMANENT, AND MAKE IT THE BASIS OF ANOTHER MODIFICATION OF THE REMAINING TENTATIVE LABELS OF THE TYPE DESCRIBED ABOVE. WHEN, AFTER AT MOST N-1 EXECUTIONS OF THE FUNDAMENTAL ITERATIVE STEP, NODE N IS PERMANENTLY LABELED, THE PROCEDURE TERMINATES.

THE PATH FROM N TO 1 MAY BE RECONSTRUCTED IF, FOR EACH LABELED NODE, THE NODE FROM WHICH IT WAS LABELED (ITS PREDECESSOR) IS RECORDED. THEN, BY STARTING AT N, WE MAY SIMPLY FOLLOW THE PREDECESSORS BACK TO THE DESTINATION NODE.

THIS PROCESS IS CORRECT, BUT FOR EASE OF HANDLING ON THE COMPUTER, CERTAIN ADJUSTMENTS HAVE BEEN MADE. THESE WERE ORIGINALLY SUGGESTED BY MINTY AND OTHERS

```
# 37   INSTEAD OF CONSIDERING A LABEL TO POSSESS A VALUE, WE SAY THAT A NODE IS EITHER   #
# 38   LABELED OR IT IS NOT, AND WE CONSIDER THE DISTANCE FROM NODE 1 TO BE A SEPARATE   #
# 39   ENTITY.  AND INSTEAD OF USING INFINITY, WE USE THE LARGEST NUMBER ALLOWABLE.      #
# 40        WE ALSO DO SOMETHING WHICH WASN'T ANTICIPATED BY THESE GENTLEMEN, AND THAT   #
# 41   IS TO SOMETIMES RUN THE ALGORITHM IN REVERSE.  NOW THE BASIC STEPS ARE STILL      #
# 42   THE SAME EXCEPT THAT WE ARE RUNNING TIME IN REVERSE, USING AN ARRIVAL TIME        #
# 43   AS THE STARTING POINT.  THUS, ALL DISTANCES ARE NEGATIVE, AND INSTEAD OF          #
# 44   LOOKING FOR THE LEAST 'LABEL', WE LOOK FOR THE GREATEST.  BY DOING THIS,          #
# 45   WE FIND OUT AT WHAT TIME A UNIT HAD TO LEAVE ITS STARTING NODE IN ORDER TO        #
# 46   ARRIVE AT ITS DESTINATION NODE AT THE GIVEN TIME.                                 #
# 47   THUS WE HAVE TWO CHOICES IN APPROACH TO THIS PROBLEM.  WE MAY WISH TO FIND        #
# 48   THE SHORTEST PATH FROM NODE 1 TO NODE N, OR WE MAY FIND WHICH NODE IS SO FAR      #
# 49   FROM NODE N.                                                                      #
# 50        RATHER THAN WORK WITH STRICTLY DISTANCE, WE WILL REALLY BE WORKING WITH      #
# 51   TIME, AND THE TWO QUESTINS WE WILL ADDRESS WILL BE 1) HOW FAST CAN WE GET FROM    #
# 52   NODE 1 TO NODE N? AND 2) GIVEN A TIME WE WISH TO ARRIVE AT NODE N, AT WHAT TIME   #
# 53   SHOULD WE LEAVE NODE 1?                                                           #
#                                                                                        #
##########################################################################################
```

COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR

```
REF
PAGE ********************************************************************************
     *                                                                              *
     *   . SOLUTION←VECTORS ARE CONSTRUCTED BY TRACING THE PATH OF NODE←VECTORS.   *
     *   .THIS IS DONE BY STARTING AT NODE I AND FOLLOWING ITS PREDECESSOR POINT-   *
     *   .ER, THEN THE PREDECESSOR'S PREDECESSOR POINTER, ETC., UNTIL NODE K IS     *
     *   .REACHED. EACH TIME A NEW PREDECESSOR IS REACHED, A NEW SOLUTION←VECTOR    *
     *   .IS CREATED. ITS APPROPRIATE VALUES ARE ENTERED, AND IT IS LINKED TO       *
     *   .ANY OTHER SOLUTION←VECTORS THAT REFER TO THE SAME NODE. AT THE SAME       *
     *   .TIME AS THE SOLUTION←VECTORS ARE BEING CONSTRUCTED, THE PATH'S ROUTE←     *
     *   .VECTOR IS ALSO BEING MADE. THIS IS BECAUSE MANY OF ITS ENTRIES ARE        *
     *   .ACCUMULATIONS OF INFORMATION GAINED AT EACH NODE.                         *
     *   . ONE WORD OF NOTE, FOR UNIFORMITY, THE SOLUTION←VECTORS OF ALL FOR-       *
     *   .WARD PATHS REMAIN AS THEY ARE CONSTRUCTED, BUT THE BACKWARD PATH SOL-     *
     *   .UTION←VECTORS HAVE THEIR ORDER REVERSED, THUS, THERE IS NO DATA BASE      *
     *   .DISTINCTION BETWEEN BACKWARD AND FORWARD PATHS.                           *
     *   .                                                                          *
     *   INCREMENT ROUTE NUMBER COUNTER                                             *
     *   SET THE UNIT NUMBER                                                        *
     *   SET TOTAL ROUTE TIME TO 0                                                  *
     *   SET TOTAL RISK MEASURE TO 0                                                *
     *   SET NUMBER OF NODES TO 1                                                   *
     *   SET ROUTE MOVEMENT TYPE                                                    *
     *   SET THE STARTING TIME                                                      *
     *   SET J TO I                                                                 *
     *   do UNTIL NODE K IS REACHED                                                 *
     *       INCREMENT SOLUTION←VECTOR COUNTER                                      *
     *       SET NODE NUMBER OF SOLUTION←VECTOR TO J                                *
     *       SET LANE NUMBER AND PARK TIME                                          *
     *       SET CODE FOR CONFLICT RESOLUTION CONSIDERATION                         *
     *       INCREMENT ROUTE DISTANCE BY LINK DISTANCE FROM J TO PREDECESSOR NODE   *
     *       INCREMENT TOTAL ROUTE TIME BY LINK TIME VALUE                          *
     *       INCREMENT TOTAL RISK BY LINK RISK VALUE                                *
     *       SET TIME IN ACCORDING TO ROUTE TYPE (FORWARD OR BACKWARD)              *
     *       SET TIME OUT ACCORDING TO ROUTE TYPE                                   *
```

```
    INCREMENT NUMBER OF SOLUTION+VECTORS
    SET J TO ITS PREDECESSOR
    if J IS NOW K
       undo UNTIL NODE K IS REACHED
    endif J IS NOW K
enddo UNTIL NODE K IS REACHED
.
.REVERSE THE ORDER OF THE SOLUTION+VECTORS IF NECESSARY
.
if MOVEMENT TYPE IS BACKWARD
    REVERSE ORDER OF SOLUTION+VECTORS
endif MOVEMENT TYPE IS BACKWARD
.
if SOLUTION+VECTOR POINTER OF NODE+VECTOR = 0
    SET SOLUTION+VECTOR POINTER TO SOLUTION+VECTOR COUNTER
    SET MULTIPLE USE OF SOLUTION+VECTOR TO -1
else POINTER DOESN'T EQUAL 0
    do FOREVER
       FOLLOW POINTER
       if MULTIPLE USE OF POINTER = -1
          SET MULTIPLE USE OF POINTER TO SOLUTION+VECTOR COUNTER
          SET MULTIPLE USE OF SOLUTION+VECTOR TO -1
          undo FOREVER
       endif MULTIPLE USE OF POINTER = -1
    enddo FOREVER
endif SOLUTION+VECTOR POINTER OF NODE+VECTOR = 0
SET ROUTE LIST HEAD POINTER
if MOVEMENT TYPE IS FORWRD
    ADD PARKING TIMES WHERE APPROPRIATE
endif MOVEMENT TYPE IS FORWARD
PROMPT FOR ENTRY OF PRINTING OPTION
if USER WANTS PRINTING
    PRINT CONTENTS OF SOLUTION+VECTORS
    PRINT ANY TIME CONFLICTS
endif USER WANTS PRINTING
return TO CALLING PROGRAM
```

CALCULATE SECOND BEST PATH

```
REF
PAGE ***********************************************************************
     *                                                                     *
     *  . THIS IS DONE BY SYSTEMATICALLY 'ERASING' NODES ALONG THE BEST PATH *
     *  .AND CALCULATING THE NEW BEST PATH. AFTER THE NEW PATH IS CALCULATED,*
     *  .THE ERASED NODE IS RESTORED, AND THE NEXT ERASED, AND SO ON. THE NEW *
     *  .PATH WITH THE COMPLETION TIME CLOSEST TO THE INITIAL BEST PATH IS THEN *
     *  .THE SECOND BEST PATH. FOR ADDED INSIGHT, WE WILL KEEP THE NEXT THREE *
     *  .BEST PATHS, HOWEVER, IT IS OFTEN THE CASE THAT TWO (OR EVEN ALL THREE) *
     *  .OF THE NEXT BEST PATHS ARE IDENTICAL.                              *
     *  .                                                                   *
     *  do UPON FUNCTION REQUEST                                            *
  54 *      PURGE ALL ROUTES                                                *
     *      .                                                               *
     *      .THIS ROUTINE CREATES THREE FICTITIOUS UNITS TO BE ASSOCIATED WITH THE *
     *      .THREE NEW CALCULATED ROUTES. CHECK FOR STORAGE ROOM IN THE UNITS DATA *
     *      .BASE.                                                          *
     *      .                                                               *
     *      if THERE IS NOT ENOUGH ROOM IN UNITS FOR THREE MORE UNIT+VECTORS *
     *          PRINT ERROR MESSAGE                                         *
     *          return TO CALLING PROGRAM                                   *
     *      endif THERE IS NOT ENOUGH ROOM IN UNITS FOR THREE MORE UNIT+VECTORS *
     *      PROMPT FOR UNIT NAME                                            *
     *      if INPUT IS NULL                                                *
     *          return TO CALLING PROGRAM                                   *
     *      endif INPUT IS NULL                                             *
     *      if ENTERED UNIT NAME IS NOT IN THE DATA BASE                    *
     *          PRINT ERROR MESSAGE                                         *
     *          TRY AGAIN                                                   *
     *      else ENTERED UNIT NAME IS LISTED                                *
     *          PROMPT FOR STARTING TIME                                    *
     *          PROMPT FOR RISK AND TIME PARAMETERS ...C AND K              *
     *          if DESTINATION IS FEBA AND NONE HAS BEEN DEFINED            *
     *              PRINT ERROR MESSAGE                                     *
     *              return TO CALLING PROGRAM                               *
```

```
          endif DESTINATION IS FEBA
          CALCULATE BEST PATH ...FORWARD MOVEMENT WITH NO DECONFLICTING
59        COMPUTE SOLUTION+VECTORS AND ROUTE+VECTOR
          .
          .NOW SYSTEMATICALLY ERASE NODES AND CALCULATE NEW PATHS
          .
          INITIALIZE ARRAY TO CONTAIN THREE NEAREST TIMES TO 32767
          do FOR SECOND THROUGH NEXT-TO-LAST NODES OF THE BEST PATH
             ERASE ROADS LEADING TO THE NODE
55           REINITIALIZE 'WORKING LIST' NODE+VECTOR ENTRIES
             CALCULATE BEST PATH ...FORWARD WITH NO DECONFLICTING
             COMPARE COMPLETION TIME WITH OTHERS IN TIME ARRAY
             if THIS TIME IS LESS THAN ANY ONE TIME IN ARRAY
                PLACE THIS TIME IN ARRAY
                ORDER ARRAY IN ASCENDING FASHION
                REMEMBER ERASED NODE
             endif THIS TIME IS LESS THAN ANY ONE TIME IN ARRAY
             RESTORE ERASED NODE'S ROADS
          enddo FOR SECOND THROUGH NEXT-TO-LAST NODES OF THE BEST PATH
          .
          .CALCULATE THE THREE NEXT BEST PATHS AND ROUTES
          .
          do FOR EACH ENTRY IN TIME ARRAY
             ERASE REMEMBERED NODE
55           REINITIALIZE 'WORKING LIST' NODE+VECTOR ENTRIES
             CALCULATE BEST PATH ...FORWARD WITH NO DECONFLICTING
59           COMPUTE SOLUTION+VECTORS AND ROUTE+VECTOR
             CREATE NEW UNIT+VECTOR
             ASSOCIATE NEW UNIT+VECTOR WITH THIS ROUTE
             PRINT MAP+NUMBER, NODE+NUMBER OF ERASED NODE
             PRINT UTM LATITUDE, LONGITUDE OF ERASED NODE
             PRINT DELAY TIME
             RESTORE ERASED NODE
          enddo FOR EACH ENTRY IN TIME ARRAY
       endif ENTERED UNIT NAME IS NOT IN THE DATA BASE
    enddo UPON FUNCTION REQUEST
```

FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL

```
REF
PAGE ****************************************************************************
     *                                                                          *
     *   . THIS PROBLEM IS: GIVEN N UNITS TRAVELING THROUGH A ROAD NETWORK, AND *
     *   .GIVEN THAT WHEN ANY TWO UNITS WISH TO USE THE SAME NODE DURING OVER-  *
     *   .LAPPING TIME INTERVALS THE ONE WITH THE HIGHER PRIORITY USES IT FIRST *
     *   .WHILE THE OTHER WAITS OR GOES AROUND, WHAT ORDERING OF PRIORITIES PRO-*
     *   .DUCES THE BEST RESULTS?                                               *
     *   . THIS ROUTINE ANSWERS THIS QUESTION BY TRYING EVERY POSSIBLE COMBINA- *
     *   .TION OF PRIORITY ORDERINGS, AND SEEING WHICH ONE DOES IN FACT PRODUCE *
     *   .THE BEST RESULTS. BY 'BEST' WE MEAN THAT THE EARLIEST START TIME OF   *
     *   .THE UNITS IN THE CHOSEN PRIORITY SCHEME IS LATER THAN ALL OTHER EAR-  *
     *   .LIEST TIMES PRODUCED USING THE OTHER PRIORITY SCHEMES.                *
     *   .                                                                      *
     *   do UPON FUNCTION REQUEST                                               *
  54 *       PURGE ALL ROUTES                                                   *
     *       PROMPT FOR TIME MOVEMENT ...FORWARD GIVES RATHER UNINTERESTING RESULTS *
     *       if MOVEMENT IS FORWARD                                             *
     *           PROMPT FOR STARTING TIME                                       *
     *       else MOVEMENT IS BACKWARD                                          *
     *           if DESTINATION IS THE FEBA AND A FEBA HAS NOT BEEN DEFINED     *
     *               PRINT ERROR MESSAGE                                        *
     *               TRY AGAIN                                                  *
     *           endif DESTINATIN IS THE FEBA                                   *
     *           PROMPT FOR ARRIVAL TIME                                        *
     *       endif MOVEMENT IS FORWARD                                          *
     *       .                                                                  *
     *       .OBTAIN UNIT NAMES. AT THIS TIME, THIS ROUTINE WILL HANDLE AT MOST FIVE *
     *       .UNITS. THIS IS BECAUSE OF TIME CONSIDERATIONS.                    *
     *       .                                                                  *
     *       do AT MOST FIVE TIMES                                              *
     *           PROMPT FOR UNIT NAME                                           *
     *           if INPUT IS NULL AND NO UNIT NAME HAS BEEN ENETERED            *
     *               return TO CALLING PROGRAM                                  *
     *           endif INPUT IS NULL AND NO UNIT NAME HAS BEEN ENTERED          *
```

```
    if INPUT IS NULL AND AT LEAST ONE UNIT NAME HAS BEEN ENTERED
        undo AT MOST FIVE TIMES
    else INPUT IS NOT NULL
        if ENTERED UNIT NAME IS NOT IN THE DATA BASE
            PRINT MESSAGE
            TRY AGAIN
        endif ENTERED UNIT NAME IS NOT IN THE DATA BASE
        REMEMBER UNIT NAME
    endif INPUT IS NULL AND AT LEAST ONE UNIT NAME HAS BEEN ENTERED
enddo AT MOST FIVE TIMES
.
.GENERATE ALL POSSIBLE PRIORITY COMBINATIONS AND MAKE PATH CALCULATIONS
.USING TIME.
.
INITIALIZE EARLIEST TIME
dc FOR EACH COMBINATION OF PRIORITIES
    RESET NODE←VECTOR SOLUTION←VECTOR POINTERS
    RESET THIS COMBINATION'S EARLIEST START TIME
    do FOR EACH UNIT
        CALCULATE BEST PATH ...WITH DECONFLICTING
        MAKE SOLUTION←VECTORS AND ROUTE←VECTOR
        if THE START TIME IS EARLIER THAN THE OTHERS
            REMEMBER THE TIME
        endif THIS PATH WORSE THAN OTHERS
    enddo FOR EACH UNIT
    if EARLIEST START TIME IS LATER THAN ALL OTHER EARLIEST TIMES
        REMEMBER PRIORITY COMBINATION
        REMEMBER EARLIEST START TIME
    endif EARLIEST START TIME IS LATER THAN ALL OTHER EARLIEST TIMES
enddo FOR EACH COMBINATION OF PRIORITIES
.
.CALCULATE RESULTS
.
REINITIALIZE SOLUTION←VECTORS AND ROUTE←VECTORS
do FOR EACH UNIT USING THE BEST PRIORITY SCHEME
    CALCULATE BEST PATH
    COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
enddo FOR EACH UNIT USING THE BEST PRIORITY SCHEME
```

```
*        PRINT MESSAGE INFORMING USER THAT RESULTS ARE AVAILABLE                    *
*    enddo UPON FUNCTION REQUEST                                                    *
*                                                                                   *
*************************************************************************************
```

CALCULATE BEST NODE AT WHICH TO INTERDICT

```
REF
PAGE
     . THIS ROUTINE CALCULATES THE NODE AT WHICH IT IS BEST TO INTERDICT,
     .WHERE 'BEST' MEANS 'HAS THE GREATEST WORTH DIFFERENCE BETWEEN THE BEST
     .PATH AND THE PATH WITH INTERDICTION'. IN ADDITION, THE PROGRAM FINDS
     .THE TIME WINDOW DURING WHICH THE INTERDICTION SHOULD OCCUR.
     . THIS IS DONE IN A MANNER SIMILAR TO THE PROCESS FOR CALCULATING THE
     .SECOND BEST PATH. EACH NODE OF THE BEST PATH IS SYSTEMATICALLY ER-
     .ASED, BUT THEN THERE IS A DIFFERENCE. THIS ROUTINE FORCES THE UNIT TO
     .TRAVEL A DISTANCE DOWN THE BEST PATH, AND THEN LETS IT GO ITS OWN WAY.
     .E.G., SUPPOSE THAT THE FIFTH NODE HAS BEEN ERASED. THERE WILL BE FOUR
     .PATHS CALCULATED FOR THIS ONE NODE ERASURE. FIRST, THE UNIT WILL BE
     .FORCED TO TRAVEL THROUGH NODE ONE, AND THEN IT MAY GO AS IT PLEASES.
     .NEXT, IT IS FORCED THROUGH NODE 2, THEN THROUGH NODE 3, AND FINALLY,
     .THROUGH NODE FOUR. IN THIS WAY, WE SIMULATE TIME RELATED INTERDICTION.
     .
     do UPON FUNCTION REQUEST
54      PURGE ALL ROUTES
        .
        .WE WILL ACTUALLY KEEP THE THREE BEST INTERDICTION CASES IN AN ARRAY.
        .AND, WE WILL CREATE THREE FICTITIOUS UNITS.
        .
        INITIALIZE BEST ARRAY
        if THERE IS NOT ROOM FOR THREE MORE UNIT←VECTORS
           PRINT MESSAGE
           return TO CALLING PROGRAM
        endif THERE IS NOT ROOM FOR THREE MOR UNIT←VECTORS
        PROMPT FOR UNIT NAME
        if INPUT IS NULL
           return TO CALLING PROGRAM
        else INPUT IS NOT NULL
           if UNIT NAME IS NOT IN DATA BASE
              PRINT MESSAGE
              TRY AGAIN
```

```
        endif UNIT NAME IS NOT IN DATA BASE
    endif INPUT IS NULL
    .
    .GET PARAMETERS
    .
    PROMPT FOR START TIME
    PROMPT FOR RISK AND TIME PARAMETERS ...C AND K
    CALCULATE BEST PATH
59  COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
    .
    .SYSTEMATICALLY ERASE NODES, AND FORCE UNIT TO TRAVEL
    .
    do FOR THE SECOND THROUGH NEXT-TO-LAST NODES OF THE BEST PATH
        ERASE NODE
        do FOR EACH NODE FROM FIRST NODE TO PREDECESSOR OF ERASED NODE
            LOOK IN SOLUTION←VECTOR OF NODE TO OBTAIN START TIME
            .
            .WE'LL ONLY CALCULATE FROM THIS NODE TO THE DESTINATION
            .
            CALCULATE BEST PATH OF THIS NODE TO THE DESTINATION
            if PATH'S WORTH IS GREATER THAN ANY OF THOSE IN THE ARRAY
                PLACE THIS WORTH IN WORTH ARRAY
                REMEMBER INTERDICTED NODE
                REMEMBER 'FORCED-TO' NODE
            endif PATH'S WORTH IS GREATER THAN ANY OF THOSE IN THE ARRAY
        enddo FOR EACH NODE FROM FIRST NODE TO PREDECESSOR OF ERASED NODE
        RESTORE ERASED NODE
    enddo FOR THE SECOND THROUGH NEXT-TO-LAST NODE OF BEST PATH
    .
    .CALCULATE CHOSEN PATHS. EACH PATH WILL BE THE COMBINATION OF TWO
    .PATHS, ONE FROM THE FIRST NODE TO THE INTERDICTED NODE, THEN ANOTHER
    .FROM THE INTERDICTED NODE TO THE DESTINATION.
    .
    do FOR EACH ENTRY IN WORTH ARRAY
        ERASE REMEMBERED NODE
        INITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES
        CALCULATE BEST PATH FROM START NODE TO PREDECESSOR OF ERASED NODE
59      COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
```

```
      * 71          INITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES                              *
      * 72          CALCULATE BEST PATH FROM PREDECESSOR TO DESTINATION NODE                   *
   59 * 73          COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR                                  *
      * 74          COMBINE SOLUTION←VECTORS AND ROUTE←VECTORS TO MAKE ONE ROUTE               *
      * 75          CREATE NEW UNIT ASSOCIATED WITH THIS ROUTE                                 *
      * 76          PRINT MAP←NUMBER,NODE←NUMBER OF INTERDICTED NODE                           *
      * 77          PRINT MAP←NUMBER, NODE←NUMBER OF 'FORCED' NODE                             *
      * 78          PRINT TIME WINDOW                                                          *
      * 79          PRINT TIME DELAY                                                           *
      * 80       enddo FOR EACH ENTRY IN WORTH ARRAY                                           *
      * 81       return TO CALLING PROGRAM                                                     *
      * 82    enddo UPON FUNCTION REQUEST                                                      *
      *                                                                                        *
      ******************************************************************************************
****** TOO MANY LINES IN SEGMENT
```

PRESENT RESULTS

```
REF
PAGE
        . THESE ROUTINES ENABLE THE USER TO SEE THE RESULTS OF THE PATH ALGO-
        .RITHMS AS PRINTED OUTPUT.
        .
        do UPON FUNCTION REQUEST
            do FOREVER
                DISPLAY FUNCTION MENU
                WAIT FOR USER INPUT
                do CASE OF
      TABLE:
 54                 PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES
      HARDCOPY:
 65                 PRINT PATH STATISTICS
      EXIT:
                    return TO CALLING PROGRAM
                enddo CASE OF
            enddo FOREVER
        enddo UPON FUNCTION REQUEST
```

PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES

REF
PAGE

```
. IF THIS FUNCTION IS REQUESTED, A TABLE SHOWING ALL ROUTE NUMBERS, EACH
.WITH THE UNIT NAME WITH WHICH IT IS ASSOCIATED, WILL BE PRINTED.
.
do UPON FUNCTION REQUEST
    PRINT TABLE OF ROUTE NUMBERS AND ASSOCIATED UNIT NAMES
    return TO CALLING PROGRAM
enddo UPON FUNCTION REQUEST
```

AD-A069 029 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/2

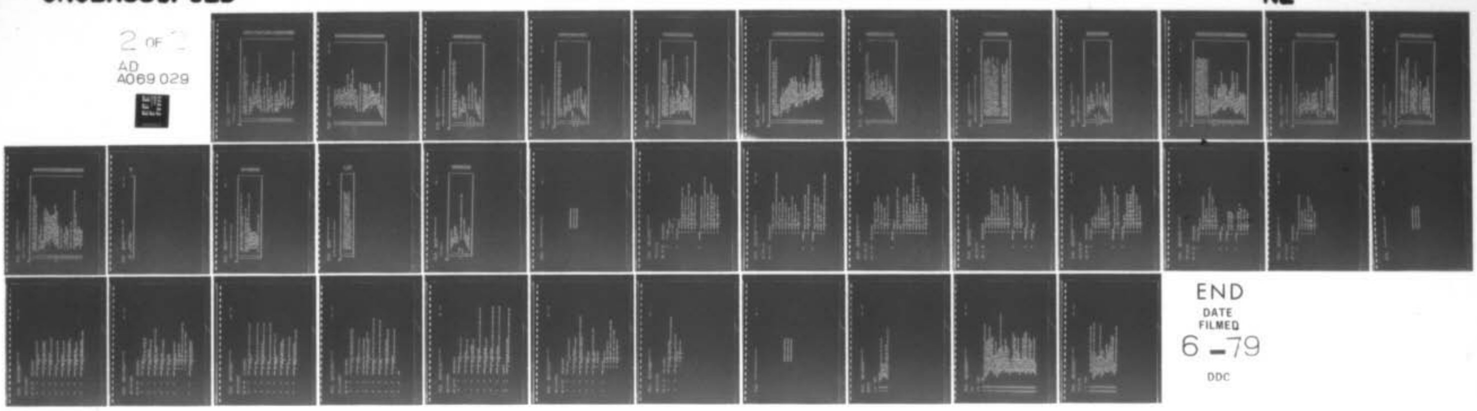
LAMAS SYSTEM MANUAL.(U)

FEB 78 DAAG39-77-C-0112

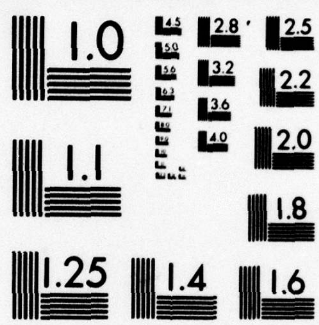
UNCLASSIFIED NL

2 OF 2

AD
A069 029



END
DATE
FILMED
6 - 79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

PRINT PATH STATISTICS

REF
PAGE

```
. THIS ENABLE THE USER TO HAVE HARD-COPY EVIDENCE OF THE CALCULATIONS
.PERFORMED BY THIS SYSTEM. THIS INCLUDES TOTAL PATH STATISTICS, AS WELL
.AS THE ABILITY TO HAVE OUTPUT IN A SNAPSHOT MODE.
.
do UPON FUNCTION REQUEST
    do FOR EACH UNIT
        if THERE IS NO ASSOCIATED ROUTE
            PRINT MESSAGE
            cycle FOR EACH UNIT
        endif THERE IS NO ASSOCIATED ROUTE
        PRINT TOTAL PATH STATISTICS ...DISTANCE, TIME, START AND FINSH, ETC.
    enddo FOR EACH UNIT
    .
    .NOW SEE ABOUT SNAPSHOT MODE
    .
    PROMPT FOR USER DESIRE TO HAVE SNAPSHOT OUTPUT
    if INPUT IS NULL
        return TO CALLING PROGRAM
    endif INPUT IS NULL
    .
    PROMPT FOR START TIME
    PROMPT FOR TIME INCREMENT
    do FOR EACH UNIT
        if THERE IS NO ASSOCIATED ROUTE
            PRINT MESSAGE
            cycle FOR EACH UNIT
        endif THERE IS NO ASSOCIATED ROUTE
        .
        .SNAPSHOT MODE
        .
        do FOREVER
            FIND UNIT POSITION FOR TIME OF INTEREST ...AS IN SNAPSHOT DISPLAY
```

```
              if UNIT HAS NOT STARTED
                 PRINT MESSAGE
                 cycle FOR EACH UNIT
              elseif UNIT HAS FINISHED
                 PRINT MESSAGE
                 cycle FOR EACH UNIT
              endif UNIT HAS NOT STARTED
              if UNIT IS AT A NODE
                 PRINT STATISTICS FOR UNIT AT THE NODE
                 cycle FOR EACH UNIT
              else UNIT IS BETWEEN TWO NODES
                 PRINT STATISTICS FOR UNIT BETWEEN THE NODES
                 cycle FOR EACH UNIT
              endif UNIT IS AT A NODE
           enddo FOR EACH UNIT
           .
           .SEE IF THE USER WISHES TO CONTINUE
           .
           PROMPT FOR USER DESIRE TO CONTINUE
           if USER DOES NOT WANT TO CONTINUE
              PROMPT FOR NEW TIME INCREMENT
              if INPUT IS NULL
                 return TO CALLING PROGRAM
              else INPUT IS NOT NULL
                 SET TIME INCREMENT
                 INCREMENT TIME
                 cycle FOREVER
              endif INPUT IS NULL
           endif USER DOES NOT WANT TO CONTINUE
        enddo FOREVER
     enddo UPON FUNCTION REQUEST
```

IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE

```
REF
PAGE
        . THIS IS THE OTHER HALF OF THE LAMAS SYSTEM. IT IS MUCH MORE LIMITED
        .IN SCOPE THAN THE ROAD NETWORK SECTION, AND THIS IS MAINLY DUE TO THE
        .NATURE OF THE DATA BASE INVOLVED.
        .
        do UPON FUNCTION REQUEST
            do FOREVER
                DISPLAY FUNCTION MENU
                WAIT FOR USER INPUT
                do CASE OF
     INITIALIZE:
67                  PERFORM TERRAIN INITIALIZATION
     PATHS:
71                  PERFORM TERRAIN PATH CALCULATIONS
     EXIT:
                    return TO CALLING PROGRAM
                enddo CASE OF
            enddo FOREVER
        enddo UPON FUNCTION REQUEST
```

PERFORM TERRAIN INITIALIZATION

```
REF
PAGE
         . IN THE CROSS-COUNTRY AND CONCEALMENT SECTION, THERE ARE ONLY TWO
         .THINGS WHICH NEED TO BE INITIALIZED, THE MAP+NUMBER, AND THE UNITS
         .DATA BASE.

         do UPON FUNCTION REQUEST
             do FOREVER
                 DISPLAY FUNCTION MENU
                 WAIT FOR USER INPUT
                 do CASE OF
      MAP+NUMBER:
68                   MAP+NUMBER TO BE USED
      UNITS:
69                   ESTABLISH UNIT+VECTORS
      EXIT:
                     return TO CALLING PROGRAM
                 enddo CASE OF
             enddo FOREVER
         enddo UPON FUNCTION REQUEST
```

MAP←NUMBER TO BE USED

REF
PAGE

```
. FOR THE CROSS-COUNTRY AND CONCEALMENT ALGORITHMS, ONLY ONE MAP MAY
.BE CONSIDERED AT ONE TIME. THIS IS SO BECAUSE OF THE LARGE NUMBER OF
.NODES WHICH ONE MAP REPRESENTS (9186), AND THE TIME INVOLVED FOR A
.PATH CALCULATION USING THAT MANY NODES.
.
do UPON FUNCTION REQUEST
    READ 'CCMDIR.DAT' INTO MEMORY ...CROSS-COUNTRY AND CONCEALMENT DIRECTORY
    PROMPT FOR MAP←NUMBER
    if INPUT IS NULL
        return TO CALLING PROGRAM
    elseif INPUT NUMBER IS NOT IN DIRECTORY
        PRINT ERROR MESSAGE
        TRY AGAIN
    else INPUT NUMBER IS IN THE DIRECTORY
        READ THE MAP'S NODES INTO MEMORY
        ESTABLISH THE SCREEN EXTREMES ...VALUES ARE IN THE DIRECTORY ENTRY
        return TO CALLING PROGRAM
    endif INPUT IS NULL
enddo UPON FUNCTION REQUEST
```

ESTABLISH UNIT+VECTORS

```
REF
PAGE ****************************************************************************
     *                                                                          *
     *  . JUST AS IN THE ROAD NETWORK SECTION, THIS DATA BASE MUST BE CON-      *
     *  .STRUCTED BY THE USER. THE ONLY DIFFERENCE BETWEEN THE TWO IS THAT      *
     *  .HERE, THE START AND DESTINATION ARE DESCRIBED AS UTM LATITUDE AND LON- *
     *  .GITUDE PAIRS.                                                          *
     *  .                                                                       *
     *  do UPON FUNCTION REQUEST                                                *
     *      if NO MAP HAS BEEN INITIALIZED                                      *
     *          PRINT ERROR MESSAGE                                             *
     *          return TO CALLING PROGRAM                                       *
     *      else A MAP HAS BEEN INITIALIZED                                     *
     *          do UNTIL UNIT+COUNTER EQUALS 60                                 *
     *              if USER DESIRES RE-INITIALIZATION                           *
     *                  CLEAR ALL EXISTING UNIT+VECTORS                         *
     *                  RESET UNIT+COUNTER TO ZERO                              *
     *              endif USER DESIRES RE-INITIALIZATION                        *
     *              PROMPT FOR ENTRY OF UNIT NAME                               *
     *              if INPUT IS NULL                                            *
     *                  return TO CALLING PROGRAM                               *
     *              else INPUT IS NOT NULL                                      *
     *                  if THIS UNIT NAME ALREADY EXISTS                        *
     *                      PRINT ERROR MESSAGE                                 *
     *                      TRY AGAIN                                           *
     *                  else UNIT NAME IS NEW                                   *
     *                      PROMPT FOR UTM COORDINATES OF STARTING NODE         *
     *                      if COORDINATES DO NOT LIE WITHIN THE INITIALIZED MAP *
     *                          PRINT ERROR MESSAGE                             *
     *                          TRY AGAIN                                       *
     *                      endif COORDINATES DO NOT LIE WITHIN THE INITIALIZED MAP *
     *                      PROMPT FOR UTM COORDINATE OF DESTINATION NODE       *
     *                      if COORDINATES DO NOT LIE WITHIN THE INITIALIZED MAP *
     *                          PRINT ERROR MESSAGE                             *
     *                          TRY AGAIN                                       *
```

```
                endif COORDINATES DO NOT LIE WITHIN THE INITIALIZED MAP
                CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE
                MAKE ASSIGNMENTS TO UNIT+VECTOR
                PROMPT FOR PRIORITY
                if PRIORITY NOT BETWEEN 0 AND 127
                    PRINT ERROR MESSAGE
                    TRY AGAIN
                endif PRIORITY NOT BETWEEN 0 AND 127
                PROMPT FOR TYPE CODE
                if TYPE CODE NOT ALLOWABLE
                    PRINT ERROR MESSAGE
                    TRY AGAIN
                endif TYPE CODE NOT ALLOWABLE
            endif THIS UNIT NAME ALREADY EXISTS
        endif INPUT IS NULL
    enddo UNTIL UNIT+COUNTER EQUALS SIXTY
  endif NO MAP HAS BEEN INITIALIZED
enddo UPON FUNCTION REQUEST
```

CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE

REF
PAGE

```
.IN ORDER TO KEEP THE MEMORY USAGE AT A MINIMUM, AN INDEXING SCHEME
.HAS BEEN DEVISED. THE DATA BASE HAS BEEN PACKED TOGETHER SO THAT
.ONE WORD CONTAINS FOUR DATA VALUES, OR EVERY DATA VALUE IS REPRESENTED
.BY FOUR BITS. ONE FOUR-BIT AREA IS CALLED A NIBBLE (HALF OF A BYTE).
.THUS, THERE ARE 8184 NIBBLES IN THE DATA BASE.
. THE POINT IS THEN TO MAP THE UTM COORDINATE PAIR TO ONE SQUARE ON THE
.93 X 88 GRID, AND THEN TRANSLATE THAT VALUE TO A NIBBLE VALUE.
.THE POSSIBLE RANGE OF THE UTM COORDINATE PAIRS IS 1 TO 506
.(22 * 23, = SQUARE KILOMETERS ON A MAP), SO WE'RE MAPPING 1 TO 506 TO
.1 TO 8184.
.
CALCULATE THE VALUE BETWEEN 1 AND 506 FOR THE GIVEN COORDINATES
CALCULATE THE PERCENTAGE OF THIS VALUE OF THE WHOLE
MULTIPLY THE PERCENTAGE BY 8183 AND ADD 1 TO OBTAIN THE NIBBLE
ASSIGN THE VALUE TO EITHER STARTING LOCATION OR DESTINATION LOCATION
```

PERFORM TERRAIN PATH CALCULATIONS

```
REF
PAGE
        do UPON FUNCTION REQUEST
            do FOREVER
                DISPLAY FUNCTION MENU
                WAIT FOR USER INPUT
                do CASE OF
    PATH:
 72                 CALCULATE CROSS-COUNTRY PATH
    SOLUTION:
 78                 DISPLAY RESULTS
    EXIT:
                    return TO THE CALLING PROGRAM
                enddo CASE OF
            enddo FOREVER
        enddo UPON FUNCTION REQUEST
```

CALCULATE CROSS-COUNTRY PATH

```
REF
PAGE
      . THIS ROUTINE CALCULATES THE PATH OF A USER-ENTERED UNIT USING THE
      .CROSS-COUNTRY DATA BASE. THE MAJOR DIFFERENCE BETWEEN THIS ROUTINE AND
      .THE ROUTINE FOR THE ROAD NETWORK IS THAT THERE IS NOW AN ARRAY CALLED
      .THE WORKING LIST. PREVIOUSLY, THE WORKING LIST VARIABLES WERE STORED
      .WITHIN THE NODE+VECTOR, BUT NOW, WITH 8184 NODES TO BE CONSIDERED, IT
      .IS TOTALLY IMPRACTICAL TO TRY TO RETAIN SPACE FOR EACH NODE'S WORKING
      .LIST ENTIRES. THUS, A WORKING LIST ARRAY EXISTS. IT IS TREATED AS A
      .LIST OF VECTORS, BEING USED IN ORDER (FROM INDEX 1 TO N) AS THEY ARE
      .NEEDED BY THE ALGORITHM.
      .
      do UPON FUNCTION REQUEST
54        PURGE ALL ROUTES
73        OBTAIN UNIT NAMES TO BE CONSIDERED
          do FOR EACH UNIT
74            OBTAIN THE PATH PARAMETERS
              if THERE IS AN ERROR RETURN
                  GO BACK TO OBTAIN UNIT NAMES AND TRY AGAIN
              endif THERE IS AN ERROR RETURN
75            CALCULATE THE CROSS-COUNTRY PATH
              if THE PATH COULD NOT COMPLETE
                  PRINT MESSAGE
                  cycle FOR EACH UNIT
              else THE PATH DID COMPLETE
77                CONSTRUCT SOLUTION+VECTORS AND ROUTE+VECTOR
              endif THE PATH COULD NOT COMPLETE
          enddo FOR EACH UNIT
          PROMPT FOR USER WISH FOR SOLUTION+VECTOR PRINTOUT
          if USER WANTS OUTPUT
              PRINT SOLUTION+VECTORS
          endif USER WANTS OUTPUT
          return TO CALLING PROGRAM
      enddo UPON FUNCTION REQUEST
```

OBTAIN UNIT NAMES TO BE CONSIDERED

REF
PAGE

```
. THE USER MAY ENTER AT MOST SIXTY NAMES.
.
do SIXTY TIMES
    PROMPT ENTRY OF UNIT NAME
    if NULL INPUT
        undo SIXTY TIMES
    endif NULL INPUT
    if INPUT NAME DOES NOT EXIST
        PRINT ERROR MESSAGE
        TRY AGAIN
    else UNIT NAME DOES EXIST
        INCREMENT UNIT COUNTER
        REMEMBER THE UNIT NAME
    endif INPUT NAME DOES NOT EXIST
enddo SIXTY TIMES
.
.ALL UNIT NAME HAVE BEEN ENTERED.
.
do ONCE
    if LESS THAN TWO UNIT NAMES HAVE BEEN ENTERED
        undo ONCE
    else MORE THAN ONE NAME HAS BEEN ENTERED
        SORT UNIT NAMES ACCORDING TO THEIR PRIORITY ...BUBBLE SORT
    endif LESS THAN TWO UNIT NAMES HAVE BEEN ENTERED
enddo ONCE
```

OBTAIN THE PATH PARAMETERS

REF
PAGE

```
do
    PROMPT FOR MOVEMENT TYPE ...BACKWARDS OR FORWARDS IN TIME
    PROMPT FOR START TIME ...DEPENDENT UPON MOVEMENT TYPE
    .
    .REINITIALIZATION OF THE WORKING LIST IS NEEDED ONLY IF EITHER THE
    .UNIT TYPE, START TIME, MOVEMENT TYPE, OR START LOCATION HAS CHANGED.
    .
    if REINITIALIZATION IS NEEDED
        SET WORKING LIST ARRAY ENTRIES TO ZERO
    endif REINITIALIZATION IS NEEDED
    ESTABLISH THE BEGINNING NODE (=K), AND THE DESTINATION NODE (=I)
    SET J TO K ...J WILL TRACE THE PATH
    .
    .SET CUMMULATIVE TIME AND WORTH FOR NODE J.
    .
    if J HAS NO WORKING LIST ENTRY
        CREATE NEW WORKING LIST ENTRY
    endif J HAS NO WORKING LIST ENTRY
    SET CUMMULATIVE TIME TO ENTERED BEGINNING TIME
    SET CUMMULATIVE WORTH TO 0
enddo
```

CALCULATE THE CROSS-COUNTRY PATH

```
REF
PAGE ***********************************************************************
     *                                                                    *
     * . GIVEN NODE J, THE LAST NODE TO BE LABELED, THIS ROUTINE CALCULATES *
     * .ITS ADJACENT NODES TIME VALUES. THEN THE NEXT NODE TO BE LABELED IS *
     * .SELECTED AND THE PROCESS CONTINUES UNTIL NODE I IS LABELED.        *
     * .                                                                  *
     * do UNTIL I IS REACHED                                              *
     *     FIND NODE J IN THE WORKING LIST                                *
     *     LABEL NODE J                                                   *
     *     do FOR EACH ADJACENT NODE OF J ...EIGHT IN ALL                 *
     *         if THE ADJACENT NODE HAS NO WORKING LIST ENTRY             *
     *             CREATE AN ENTRY FOR THE ADJACENT NODE                  *
     *         else THE NODE HAS A WORKING LIST ENTRY                     *
     *             if THE ADJACENT NODE IS LABELED                        *
     *                 cycle FOR EACH ADJACENT NODE                       *
     *             endif THE ADJACENT NODE IS LABELED                     *
     *         endif THE ADJACENT NODE HAS NO WORKING LIST ENTRY          *
  75 *         CALCULATE THE ADJACENT NODE'S TIME VALUES                  *
     *     enddo FOR EACH ADJACENT NODE                                   *
     *     .                                                              *
     *     .LOOK FOR I BEING LABELED                                      *
     *     .                                                              *
     *     if I IS NOW LABELED                                            *
     *         undo UNTIL I IS REACHED                                    *
     *     endif IS IS NOW LABELED                                        *
     *     .                                                              *
     *     .CALCULATE THE NEXT NODE TO LABEL                              *
     *     .                                                              *
     *     FIND THE UNLABELED WORKING LIST ENTRY WHCIH HAS THE LEAST WORTH MEASURE *
     *     if NONE EXISTS ...THE ROUTE COULDN'T FINISH                    *
     *         PRINT ERROR MESSAGE                                        *
     *         undo UNTIL I IS REACHED                                    *
     *     else AN ENTRY WAS FOUND                                        *
     *         SET J TO THIS ENTRY                                        *
```

```
* 33       endif NONE EXISTS                                        *    2214
* 34    enddo UNTIL I IS REACHED                                    *    2215
*                                                                   *
*********************************************************************
```

CALCULATE THE ADJACENT NODE'S TIME VALUES

REF
PAGE

```
. AGAIN THIS IS SIMILAR TO THE TIME ROUTINE FOR THE ROAD NETWORK, AND
.AGAIN THE MAIN DIFFERENCE IS THE DATA BASE.
.
CALCULATE THE TIME OF TRAVEL BETWEEN J AND THIS NODE
CALCULATE CUMMULATIVE TIME FOR THIS LINK
if THE CUMMULATIVE TIME CALCULATED IS LESS THAN THE WORKING LIST ENTRY
    SET NEW CUMMULATIVE TIME
    SET NEW TIME MEASURE
    SET PREDECESSOR NODE
    SET CUMMULATIVE WORTH
    SET WORTH MEASURE
endif THE CUMMULATIVE TIME CALCULATED IS LESS
```

CONSTRUCT SOLUTION+VECTORS AND ROUTE+VECTOR

```
REF
PAGE ************************************************************************************
     *                                                                                  *
     *  1    . THIS ROUTINE IS IDENTICAL TO THE ROAD NETWORK ROUTINE WHICH PERFORMS     *
     *  2    .THE SAME TASK EXCEPT THAT THE WORKING LIST IS REFERENCED IN ORDER TO     *
     *  3    .OBTAIN CERTAIN VALUES, SUCH AS 1) TOTAL WORTH MEASURE, 2) STARTING        *
     *  4    .TIME, 3) TIME MEASURE (LINK TIME), 4) PREDECESSOR'S CUMMULATIVE TIME,     *
     *  5    .AND 5) THIS NODE'S CUMMULATIVE TIME                                       *
     *                                                                                  *
     ************************************************************************************
```

DISPLAY RESULTS

REF PAGE

```
    do UPON FUNCTION REQUEST
        do FOREVER
            DISPLAY FUNCTION MENU
            WAIT FOR USER INPUT
            do CASE OF
TABLE:
64              PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES
EXIT:
                return TO CALLING PROGRAM
            enddo CASE OF
        enddo FOREVER
    enddo UPON FUNCTION REQUEST
```

```
************************
*                      *
* INDEX TO DATA ITEMS  *
*                      *
************************
```

```
INDEX TO DATA ITEMS
-------------------

PAGE  LINE  TYPE       NAME AND REFERENCES
----  ----  ----       -------------------

             DI        MAIN<-PROGRAM
                               26   PREPARATION
                                          1
                               35   LAMAS
                                          1    15

             DI        MAP<-DIRECTORY
                               26   PREPARATION
                                          2
                               27   CREATE NODE<-VECTOR AND MAP<-DIRECTORY FILES ON DISK
                                         21   43   45
                               32   CREATE MAP<-DIRECTORY ENTRY FOR THIS MAP
                                          8    9
                               39   ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE<-VECTORS
                                         12   16   17   26
                               40   READ THIS MAP'S NODES INTO MAIN MEMORY
                                         18   19   20
                               41   CHANGE ADJACENT NODE'S MAP<-NUMBER TO INDEX
                                         16

             DI        MAP<-NUMBER
                               32   CREATE MAP<-DIRECTORY ENTRY FOR THIS MAP
                                          8
                               33   CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK
                                         50
                               38   INITIALIZE MAP<-NUMBERS AND SCREEN EXTREMES
                                          5    6    7
                               39   ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE<-VECTORS
                                         15   16   21   22   24   26   27
                               41   CHANGE ADJACENT NODE'S MAP<-NUMBER TO INDEX
                                         16
                               43   ESTABLISH A FORWARD EDGE OF BATTLE AREA
                                          5    9   14
```

INDEX TO DATA ITEMS

```
PAGE  LINE  TYPE        NAME AND REFERENCES
----  ----  ----        -------------------
                              45   PERFORM INTERDICTIVE OPERATIONS
                                        11
                              47   PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINAT
                                        17
                              48   CHANGE CONTENTS OF A NODE←VECTOR
                                        7   12
                              50   DELETE A NODE FROM THE NODE←VECTOR ARRAY
                                        9
                              51   PRINT CONTENTS OF A NODE←VECTOR
                                        2
                              60   CALCULATE SECOND BEST PATH
                                        62
                              62   CALCULATE BEST NODE AT WHICH TO INTERDICT
                                        76   77
                              67   PERFORM TERRAIN INITIALIZATION
                                        10   11
                              68   MAP←NUMBER TO BE USED
                                        8

            DI          MAP←NUMBERS
                              37   PREPARE FOR ROAD NETWORK PATH CALCULATIONS
                                        7
                              39   ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS
                                        13   14

            DI          NODE←NUMBER
                              41   CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX
                                        16
                              43   ESTABLISH A FORWARD EDGE OF BATTLE AREA
                                        5    9    14
                              46   PERFORM INTERDICTIVE OPERATIONS
                                        11
                              47   PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINAT
                                        17
```

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

INDEX TO DATA ITEMS

```
PAGE  LINE  TYPE      NAME AND REFERENCES
----  ----  ----      ---- --- ----------

            DI        ROUTE<VECTOR
                           53   CALCULATE PATH
                                     15
                           54   PURGE ALL ROUTES
                                     6    8
                           60   CALCULATE SECOND BEST PATH
                                     35   59
                           61   FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
                                     53   69
                           62   CALCULATE BEST NODE AT WHICH TO INTERDICT
                                     41   70   73
                           72   CALCULATE CROSS-COUNTRY PATH
                                     24

            DI        ROUTE<VECTORS
                           61   FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
                                     66
                           62   CALCULATE BEST NODE AT WHICH TO INTERDICT
                                     74

            DI        SOLUTION<VECTOR
                           54   PURGE ALL ROUTES
                                     9    11   14
                           56   CALCULATE AND ASSIGN TIME AND WORTH VALUES
                                     40   42   44   46   49   67
                           59   COMPUTE SOLUTION<VECTORS AND ROUTE<VECTOR
                                     24   25   46   47   47   48   53   54   58
                           61   FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
                                     49
                           62   CALCULATE BEST NODE AT WHICH TO INTERDICT
                                     48
                           72   CALCULATE CROSS-COUNTRY PATH
                                     27
```

INDEX TO DATA ITEMS

```
PAGE  LINE  TYPE     NAME AND REFERENCES
----  ----  ----     -------------------

             DI      SOLUTION←VECTORS
                             53   CALCULATE PATH
                                       15
                             59   COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
                                       33   43   65
                             60   CALCULATE SECOND BEST PATH
                                       35   59
                             61   FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
                                       53   66   69
                             62   CALCULATE BEST NODE AT WHICH TO INTERDICT
                                       41   70   73   74
                             72   CALCULATE CROSS-COUNTRY PATH
                                       24   29

             DI      TERRAIN←MODEL
                             35   LAMAS
                                       8

             DI      UNIT←COUNTER
                             42   INITIALIZE UNIT←VECTORS
                                       9    14   26   58
                             69   ESTABLISH UNIT←VECTORS
                                       11   14   48

             DI      UNIT←VECTOR
                             42   INITIALIZE UNIT←VECTORS
                                       27   33   34   40   46   52
                             52   CHANGE PROPERTIES OF A UNIT
                                       10
                             60   CALCULATE SECOND BEST PATH
                                       60   61
                             69   ESTABLISH UNIT←VECTORS
                                       35
```

INDEX TO DATA ITEMS

```
PAGE  LINE  TYPE      NAME AND REFERENCES
----  ----  ----      -------------------

             DI       UNIT+VECTORS
                              37   PREPARE FOR ROAD NETWORK PATH CALCULATIONS
                                          9
                              42   INITIALIZE UNIT+VECTORS
                                          8
                              60   CALCULATE SECOND BEST PATH
                                         16   19
                              62   CALCULATE BEST NODE AT WHICH TO INTERDICT
                                         22   25
                              67   PERFORM TERRAIN INITIALIZATION
                                         13
                              69   ESTABLISH UNIT+VECTORS
                                         13
```

```
***************************
*                         *
* INDEX TO FLOW SEGMENTS  *
*                         *
***************************
```

INDEX TO FLOW SEGMENTS

PAGE	LINE	TYPE	NAME AND REFERENCES
56		FS	CALCULATE AND ASSIGN TIME AND WORTH VALUES 53 CALCULATE PATH 08
62		FS	CALCULATE BEST NODE AT WHICH TO INTERDICT 45 PATH DETERMINATION AND DISPLAY 32
72		FS	CALCULATE CROSS-COUNTRY PATH 71 PERFORM TERRAIN PATH CALCULATIONS 7
53		FS	CALCULATE PATH 45 PATH DETERMINATION AND DISPLAY 26
60		FS	CALCULATE SECOND BEST PATH 45 PATH DETERMINATION AND DISPLAY 28
76		FS	CALCULATE THE ADJACENT NODE'S TIME VALUES 75 CALCULATE THE CROSS-COUNTRY PATH 16
75		FS	CALCULATE THE CROSS-COUNTRY PATH 72 CALCULATE CROSS-COUNTRY PATH 19
41		FS	CHANGE ADJACENT NODE'S MAP+NUMBER TO INDEX 38 INITIALIZE MAP+NUMBERS AND SCREEN EXTREMES 6

INDEX TO FLOW SEGMENTS

```
PAGE  LINE  TYPE     NAME AND REFERENCES
----  ----  ----     -------------------

 49          FS      CHANGE CHARACTERISTIC
                           48   CHANGE CONTENTS OF A NODE+VECTOR
                                  25   27   29   31   33   35

 48          FS      CHANGE CONTENTS OF A NODE+VECTOR
                           46   PERFORM INTERDICTIVE OPERATIONS
                                  13

 70          FS      CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE
                           69   ESTABLISH UNIT+VECTORS
                                  34

 52          FS      CHANGE PROPERTIES OF A UNIT
                           46   PERFORM INTERDICTIVE OPERATIONS
                                  19

 57          FS      COMPUTE NEXT NODE TO LABEL
                           53   CALCULATE PATH
                                  12

 59          FS      COMPUTE SOLUTION+VECTORS AND ROUTE+VECTOR
                           53   CALCULATE PATH
                                  15
                           60   CALCULATE SECOND BEST PATH
                                  35   59
                           61   FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
                                  69
                           62   CALCULATE BEST NODE AT WHICH TO INTERDICT
                                  41   70   73

 77          FS      CONSTRUCT SOLUTION+VECTORS AND ROUTE+VECTOR
                           72   CALCULATE CROSS-COUNTRY PATH
                                  24
```

INDEX TO FLOW SEGMENTS

```
PAGE  LINE  TYPE       NAME AND REFERENCES
----  ----  ----       -------------------

 33          FS        CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK
                               26   PREPARATION
                                          3

 30          FS        CREATE LINK CARD
                               27   CREATE NODE+VECTOR AND MAP+DIRECTORY FILES ON DISK
                                          10

 32          FS        CREATE MAP+DIRECTORY ENTRY FOR THIS MAP
                               27   CREATE NODE+VECTOR AND MAP+DIRECTORY FILES ON DISK
                                          43

 28          FS        CREATE NODE CARD
                               27   CREATE NODE+VECTOR AND MAP+DIRECTORY FILES ON DISK
                                          7

 27          FS        CREATE NODE+VECTOR AND MAP+DIRECTORY FILES ON DISK
                               26   PREPARATION
                                          2

 50          FS        DELETE A NODE FROM THE NODE+VECTOR ARRAY
                               46   PERFORM INTERDICTIVE OPERATIONS
                                          15

 78          FS        DISPLAY RESULTS
                               71   PERFORM TERRAIN PATH CALCULATIONS
                                          9

 43          FS        ESTABLISH A FORWARD EDGE OF BATTLE AREA
                               37   PREPARE FOR ROAD NETWORK PATH CALCULATIONS
                                          11
```

INDEX TO FLOW SEGMENTS

INDEX TO FLOW SEGMENTS

```
INDEX TO FLOW SEGMENTS
------------------------

PAGE  LINE  TYPE      NAME AND REFERENCES
----  ----  ----      -------------------

 63         FS        PRESENT RESULTS
                              45   PATH DETERMINATION AND DISPLAY
                                        34

 51         FS        PRINT CONTENTS OF A NODE+VECTOR
                              46   PERFORM INTERDICTIVE OPERATIONS
                                        17

 47         FS        PRINT MAP+NUMBER, NODE+NUMBER OF NODE+VECTOR NEAREST GIVEN COORDINATES
                              46   PERFORM INTERDICTIVE OPERATIONS
                                        11

 65         FS        PRINT PATH STATISTICS
                              63   PRESENT RESULTS
                                        12

 64         FS        PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES
                              63   PRESENT RESULTS
                                        10
                              78   DISPLAY RESULTS
                                        7

 54         FS        PURGE ALL ROUTES
                              53   CALCULATE PATH
                                        26
                              50   CALCULATE SECOND BEST PATH
                                        10
                              61   FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
                                        13
                              62   CALCULATE BEST NODE AT WHICH TO INTERDICT
                                        16
                              72   CALCULATE CROSS-COUNTRY PATH
                                        12
```

INDEX TO FLOW SEGMENTS

```
PAGE  LINE  TYPE      NAME AND REFERENCES
----  ----  ----      -------------------

 40          FS       READ THIS MAP'S NODES INTO MAIN MEMORY
                              39   ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE+VECTORS
                                         20

 55          FS       REINITIALIZE 'WORKING LIST' NODE+VECTOR ENTRIES
                              53   CALCULATE PATH
                                         78   83
                              60   CALCULATE SECOND BEST PATH
                                         42   57
```

```
*****************************
*                           *
* SEGMENT REFERENCE TREES   *
*                           *
*****************************
```

PREPARATION

```
LN  DEF   SEGMENT
--  ---   -------

 1   26    PREPARATION
 2   27       CREATE NODE←VECTOR AND MAP←DIRECTORY FILES ON DISK
 3   28          CREATE NODE CARD
 4   30          CREATE LINK CARD
 5   32          CREATE MAP←DIRECTORY ENTRY FOR THIS MAP
 6   33       CREATE CROSS-COUNTRY, CONCEALMENT, AND DIRECTORY FILES ON DISK
```

LAMAS

```
LN  DEF    SEGMENT
--  ---    -------

 1   35     LAMAS
 2   36        IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE NODE←VECTOR DATA BASE
 3   37           PREPARE FOR ROAD NETWORK PATH CALCULATIONS
 4   38              INITIALIZE MAP←NUMBERS AND SCREEN EXTREMES
 5   39                 ESTABLISH DIRECTORY IN MEMORY AND READ IN NODE←VECTORS
 6   40                    READ THIS MAP'S NODES INTO MAIN MEMORY
 7   41                 CHANGE ADJACENT NODE'S MAP←NUMBER TO INDEX
 8   42              INITIALIZE UNIT←VECTORS
 9   43              ESTABLISH A FORWARD EDGE OF BATTLE AREA
10   45           PATH DETERMINATION AND DISPLAY
11   46              PERFORM INTERDICTIVE OPERATIONS
12   47                 PRINT MAP←NUMBER, NODE←NUMBER OF NODE←VECTOR NEAREST GIVEN COORDINATES
13   48                 CHANGE CONTENTS OF A NODE←VECTOR
14   49                    CHANGE CHARACTERISTIC
15   50                 DELETE A NODE FROM THE NODE←VECTOR ARRAY
16   51                 PRINT CONTENTS OF A NODE←VECTOR
17   52                 CHANGE PROPERTIES OF A UNIT
18   53              CALCULATE PATH
19   54                 PURGE ALL ROUTES
20   55                 REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES
21   56                 CALCULATE AND ASSIGN TIME AND WORTH VALUES
22   57                 COMPUTE NEXT NODE TO LABEL
23   59                 COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
24   60              CALCULATE SECOND BEST PATH
25   54                 PURGE ALL ROUTES
26   59                 COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
27   55                 REINITIALIZE 'WORKING LIST' NODE←VECTOR ENTRIES
28   61              FIND THE ORDER OF PRIORITIES FOR BEST NETWORK TRAVEL
29   54                 PURGE ALL ROUTES
30   59                 COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
31   62              CALCULATE BEST NODE AT WHICH TO INTERDICT
32   54                 PURGE ALL ROUTES
33   59                 COMPUTE SOLUTION←VECTORS AND ROUTE←VECTOR
34   63              PRESENT RESULTS
```

LAMAS (CONTINUED)

```
LN  DEF    SEGMENT
--  ---    -------

35  64              PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES
36  65              PRINT PATH STATISTICS
37  66        IMPLEMENT ALGORITHMS AND FUNCTIONS USING THE CROSS-COUNTRY DATA BASE
38  67          PERFORM TERRAIN INITIALIZATION
39  68            MAP+NUMBER TO BE USED
40  69            ESTABLISH UNIT+VECTORS
41  70              CHANGE ENTERED COORDINATES TO AN INDEX INTO THE DATA BASE
42  71          PERFORM TERRAIN PATH CALCULATIONS
43  72            CALCULATE CROSS-COUNTRY PATH
44  54              PURGE ALL ROUTES
45  73              OBTAIN UNIT NAMES TO BE CONSIDERED
46  74              OBTAIN THE PATH PARAMETERS
47  75              CALCULATE THE CROSS-COUNTRY PATH
48  76                CALCULATE THE ADJACENT NODE'S TIME VALUES
49  77              CONSTRUCT SOLUTION+VECTORS AND ROUTE+VECTOR
50  78            DISPLAY RESULTS
51  64              PRINT TABLE OF ROUTE NUMBERS WITH ASSOCIATED UNIT NAMES
```